

XML JOURNAL

The World's Leading XML Resource

Volume: 1 Issue: 1 Premier Issue

XML-JOURNAL.COM

Announcing... XML DevCon 2000
Coming June 25-28, 2000

JavaCON 2000
September 24-27, 2000

FROM THE EDITOR
Welcome to XML-Journal!
by Ajit Sagar pg. 3

BOOK REVIEW
Dive into the XML Specification
Reviewed by Tija Ragas pg. 28

SYS-CON RADIO
Interview with Coco Jaenicke of eXcelon Corporation pg. 38

XML DEMYSTIFIED
Replace DTDs? Why?
by Robert DuCharme pg. 40

2B OR NOT 2B
XML: It's the 'X' that Matters
by Coco Jaenicke pg. 52

XML AND E-COMMERCE
Business-to-Business E-Commerce: XML's Killer App
by John Spiers pg. 56

IMHO
Meaning, Not Markup
by Simon Phipps pg. 66

SYS-CON MEDIA

PRESENTING XML TO THE WEB

One of many approaches to developing Web-based applications

by Bhaven Shah
see page 18

EAI: Extensible Integration of the Enterprise and Beyond *XML's self-documenting design and ability to store and communicate metadata help preserve a flexible, open-applications architecture* **6**



John Evdemon

Feature: Wireless Markup Language *Using WML to support the presentation layer of your application's architecture* **10**



Ian Moraes

Industry Insider: OASIS and XML.org *How XML industry specifications are being developed* **16**



Bob Sutor

XML Middleware: XML and Messaging *Process data in a simple and straightforward manner* **24**



Sandeep M. Nayak

Java and XML: The Promised Land *Working hand in hand to allow processing of information from external systems* **30**



Israel Hilerio

<e-BizML>: XML in the Enterprise *Some business areas in which XML is making an impact* **36**



Ajit Sagar

Feature: Building Distributed Applications with CORBA and XML *Can these technologies benefit from each other?* **42**



Nick Simha & Dermot Russell

XML Modeling: Specification for the Metadata Interchange *Bringing consistency and compatibility to applications* **58**



Ilango Kumaran S

Softquad

www.softquad.com



EDITORIAL ADVISORY BOARD

COCO JAENICKE, SIMON PHIPPS, AJIT SAGAR, BOB SUTOR

EDITOR-IN-CHIEF: AJIT SAGAR
 EXECUTIVE EDITOR: M'LOU PINKHAM
 ART DIRECTOR: ALEX BOTERO
 PRODUCTION EDITOR: CHERYL VAN SISE
 ASSOCIATE EDITOR: NANCY VALENTINE
 XML INDUSTRY NEWS EDITOR: ALAN WILLIAMSON
 E-BUSINESS EDITOR: ISRAEL HILERIO
 JAVA TECHNOLOGY EDITOR: JASON WESTRA

WRITERS IN THIS ISSUE

BOB DUCHARME, JOHN EVDEMON, ISRAEL HILERIO, COCO JAENICKE,
 ILANGO KUMARAN S, IAN MORAES, SANDEEP M. NAVAK, SIMON
 PHIPPS, TIJA RAGAS, DERMOT RUSSELL, AJIT SAGAR, BHAVEN SHAH,
 NICK SIMHA, JOHN SPIERS, BOB SUTOR

SUBSCRIPTIONS

FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS,
 PLEASE SEND YOUR LETTERS TO SUBSCRIPTION DEPARTMENT

SUBSCRIPTION HOTLINE

800 513-7111

COVER PRICE: \$8.99/ISSUE

DOMESTIC: \$49.99/YR. (6 ISSUES) CANADA/MEXICO: \$59.99/YR.

ALL OTHER COUNTRIES \$69.99

(U.S. BANKS OR MONEY ORDERS)

PUBLISHER, PRESIDENT AND CEO: FIJAT A. KIRCAALI
 VICE PRESIDENT, PRODUCTION: JIM MORGAN
 VICE PRESIDENT, MARKETING: CARMEN GONZALEZ
 CHIEF FINANCIAL OFFICER: ELI HOROWITZ
 ACCOUNTING MANAGER: JO-ANN COFFEY
 CIRCULATION MANAGER: MARY ANN MCBRIDE
 ADVERTISING ACCOUNT MANAGERS: MEGAN RING
 ROBYN FORMA
 JD.JSTORE.COM: JACLYN REDMOND
 ADVERTISING ASSISTANT: CHRISTINE RUSSELL
 GRAPHIC DESIGN INTERN: AARATHI VENKATARAMAN
 WEBMASTER: ROBERT DIAMOND
 WEB EDITOR: BARD DEMA
 WEB SERVICES CONSULTANT: BRUNO Y. DECAUDIN
 WEB SERVICES INTERN: BRYAN KREMKAU
 CUSTOMER SERVICE: CAROL KILDUFF
 ANN MARIE MILILLO
 ONLINE CUSTOMER SERVICE: AMANDA MOSKOWITZ

EDITORIAL OFFICES

SYS-CON PUBLICATIONS, INC.
 39 E. CENTRAL AVE., PEARL RIVER, NY 10965
 TELEPHONE: 914 735-7300 FAX: 914 735-6547
 SUBSCRIBE@SYS-CON.COM

XML-JOURNAL (ISSN# PENDING)
 is published bimonthly (6 times a year) by
 SYS-CON Publications, Inc., 39 E. Central Ave.,
 Pearl River, NY 10965-2306
 Periodicals Postage rates are paid at
 Pearl River, NY 10965 and additional mailing offices.
 POSTMASTER: Send address changes to:
 XML-JOURNAL, SYS-CON Publications, Inc.,
 39 E. Central Ave., Pearl River, NY 10965-2306.

@COPYRIGHT

Copyright © 2000 by SYS-CON Publications, Inc. All rights reserved.
 No part of this publication may be reproduced or transmitted in any form or by any
 means, electronic or mechanical, including photocopy or any information storage and
 retrieval system, without written permission. For promotional reprints, contact reprint
 coordinator, SYS-CON Publications, Inc., reserves the right to revise, republish and
 authorize its readers to use the articles submitted for publication.

WORLD DISTRIBUTION

CURTIS CIRCULATION COMPANY

739 RIVER ROAD, NEW MILFORD, NJ 07646-3048 PHONE: 201 634-7400

All brand and product names used on these pages are trade names,
 service marks or trademarks of their respective companies.
 SYS-CON Publications, Inc., is not affiliated with the companies
 or products covered in XML-Journal.



WRITTEN BY AJIT SAGAR EDITOR-IN-CHIEF

from
 the
 editor
 in
 the
 editor

Welcome to XML-Journal!

I'm sitting at my desk writing this editorial for the premier issue of *XML-Journal* for just one reason. And you're holding the issue in your hands for exactly the same reason: we both want to talk about XML – a technology that has revolutionized electronic commerce and enterprise computing and is going to completely refurbish the face of business as we know it today. The editorial board of *XML-Journal* and *SYS-CON*'s writers and editors want to share their experiences in using this technology with you, and we want you to share yours with us. Tell us what you think of *XML-J* and how we can best serve your needs. We want to become your one-stop shop for all matters XML.

Before I go on, I guess I should introduce myself and talk about the origins of *XML-Journal*. I've been a contributing editor to our sister publication, *Java Developer's Journal*, since its inception, and currently write an e-Java column that focuses on Java and e-business. During JavaOne last June, I had an early morning breakfast session with *SYS-CON* management and the editorial group about a new, XML-focused publication. We decided to test industry interest by publishing the September '99 *DDJ* as an XML Focus issue. This issue sold out on the newsstands and we received very positive feedback from our readers. Thus *XML-J* was born.

XML DevCon 2000: XML Conference and Expo

The feedback and industry interest indicated a lack of good, convenient and easily accessible sources of current information on XML and related technologies. For this reason *XML-J* recently announced XML DevCon 2000, the only XML conference/expo on the East Coast this year, from June 25 to 28. We are determined to make this conference the most valuable and informative XML-focused gathering ever put together. More information on the conference is available elsewhere in this issue and at www.xmldevcon2000.com. Mark your calendars!



Coming Up in XML-Journal

Now let's talk about what we're going to talk about in *XML-Journal* in the coming year. This premier issue is being published in the first quarter of the new millennium. Technologies from the last millennium will play a pivotal role in defining business in the years to come. XML is one of these technologies. Only a few years old, it already holds the promise of uniting computing platforms through a common paradigm. An interesting thing to note about the XML community is that it's a hybrid of the development and business communities. That partially explains why this magazine, unlike the others in the *SYS-CON* family, doesn't have the word *Developer's* in it. *XML-J* will concentrate on e-business and e-commerce, as well as on the development aspects of XML and related technologies.

Each issue we'll offer feature articles on a particular sector of XML in the industry. Other sections of the magazine will be dedicated to different XML technologies and verticals. *XML Standards* and *XML Insider* will bring you the latest developments in the XML language itself. Bob Sutor, chief strategy officer at OASIS, will update you on this subject in every issue. Coco Jaenike will tell you about XML in the B2B marketplace. Israel Hilerio will show you how the worlds of XML and Java meet. My own contribution will be on XML's impact on e-business.

Each month we'll also try to bring you interviews with industry leaders in XML technologies. This month we talked to Coco Jaenike about the XML-related developments at eXcelon Corporation.

In forthcoming issues we'll highlight a couple of vendors who will write about XML technologies and what role their company plays in the industry. We'll also bring you a mix of tutorials, case studies, book and product reviews, letters from readers and the latest news in the XML arena.

We'd like you to contribute your own ideas to the magazine. Have you thought of a new way to apply the technology? Do you have some insights you can share with other readers? Are there concepts you can explain to others who are trying to get aboard this fast technology train? We're eager to hear from you.

In addition to the print issue you're reading, *XML-J* is also available online at www.sys-con.com/xml/index2.html. I encourage you to submit your ideas, feedback and opinions, and to share your experiences with readers like yourselves.

So go ahead. Turn the page. And welcome to the wonderful world of *XML-J*



AUTHOR BIO

Ajit Sagar is the founding editor and editor-in-chief of XML-Journal and a member of a leading e-commerce firm in Dallas, Texas, focusing on Web-based e-commerce applications and architectures. He is a leading Java and XML expert.



XML's self-documenting design and ability to store and communicate metadata help preserve a flexible, open-applications architecture

Extensible Integration of the Enterprise and Beyond

One of the most significant challenges that businesses have traditionally faced is the integration of various system components throughout the enterprise. Over the past year, EAI (enterprise application integration) has emerged as a popular approach for integrating systems and gaining a strategic advantage in the marketplace. A corporation that has successfully integrated their internal systems is better positioned to take advantage of the opportunities that emerge following this effort. A financial services firm, for example, could profit from "cross-selling" products and services that in the past may have been tied to monolithic, proprietary systems that were incapable of sharing information. Tool vendors such as NEON, IBM and BEA have enabled businesses to take advantage of EAI using technologies such as message brokering, MOM, CORBA, COM and others.

The purpose of this article is to suggest a broader EAI strategy that leverages the power and flexibility of Extensible Markup Language. XML provides a platform-agnostic, technology-neutral form of structuring messages. By avoiding specific platform or technical messaging requirements, the approach to systems integration becomes more open and thus more extensible.

This article also examines possible approaches for using XML with database middleware, distributed objects, message-oriented middleware, application servers and EDI.

Issues Associated with EAI

TIGHTLY COUPLED DESIGNS

One of the potential issues associated with EAI is its limited scope – EAI is concerned with a single enterprise. The success of the business-to-business (B2B) e-commerce model has required firms to focus beyond their internal systems. Businesses interested in automating their supply chain via an extranet or communicating with trading partners via well-defined B2B interfaces are confronted with much more challenging integration issues than the typical EAI project. For example, a company's middleware or messaging product may not be fully compatible with all trading partners. (Promotion and adoption of a B2B

strategy is most successful when partners' technical demands are minimized or effectively eliminated.) Since EAI typically focuses on integrating applications within a single company, a data-level or API-level approach is usually chosen.

A *data-level* approach to EAI focuses on the processes, techniques and technologies required for transporting system data from one data store to another. (The term *data store* is used here instead of *database* since some EAI efforts utilize both relational and nonrelational data.) Typical processes may be devoted to extraction, transformation and loading of the data (commonly referred to as the *ETL* model). Extraction of the data typically requires some form of middleware (such as ODBC), while the transformation and loading of the data can be accomplished via custom software and/or COTS (commercial, off-the-shelf) packages.

An *API-level* approach to EAI requires the use of programmable interfaces exposed by system vendors that can be utilized to develop third-party extensions and system integration. While this approach to EAI has the advantage of using proven, mature technologies (vendor APIs are usually very reliable), the functionality of the interfaces varies depending on the vendor. The costs associated with hiring and retaining

developers familiar with vendor APIs may also be somewhat prohibitive.

While there are several advantages to using a data-level or API-level approach to EAI, the end result is usually a tightly coupled design that can be extremely difficult to extend to partners outside the enterprise. However, if the design requires all messages to be constructed in a platform-neutral format using XML, it becomes much more flexible and significantly lowers the entry barriers for partners outside the enterprise.

TOOL IMMATURITY

As stated earlier, most EAI vendors tend to focus on and promote the technical capabilities of their products (especially in terms of middleware) instead of building expertise in specific business processes and business/application semantics. Eventually they will begin to build expertise in these areas (rather than the one-size-fits-all approach currently favored by many vendors). For now, EAI vendors continue to focus on technical features since they're easily marketed and understood. A more nebulous feature (such as business semantics) can be extremely difficult to market and implement.

The emergence and subsequent popularity of XML helps facilitate the development and integration of business and application semantics – enterprises can define their own data elements (e.g., tags) to better communicate the "meaning" of their data. XML's self-documenting design enables external partners to communicate with a greater level of clarity, ensuring that message constructs will be processed in the appropriate manner.

MESSAGE BROKERS

Message brokers are frequently referred to when discussing EAI vendors and tools. Unfortunately, their function usually isn't well understood. They behave much like message-oriented middleware products (such as MQ-Series) with some additional capabilities. These additional capabilities vary with the associated vendor but usually consist of:

- Transformation options
- Store and forward messaging
- Bridges for communicating with specialized COTS packages
- Management tools for tuning the transformation rules engine and monitoring overall performance

A listing of message broker vendors and products appears elsewhere in this article.

While message brokers do a good job of connecting applications via messaging, they don't adequately communicate the semantics of the message being transmitted. As indicated previously, XML should be used to format messages for transmission since it preserves the semantics. Some message brokering tools, such as MQSeries Integrator, already offer limited support for XML (messages can be transformed by the tool's rules engine into XML using an agreed-upon schema).

Business Processes and the Hub/Spoke Architecture

A typical EAI solution employs a hub-and-spoke architecture as illustrated in Figure 1.

Applications within the enterprise access data from a central hub via application spokes. Data within the hub is populated via ETL processes that can be colocated at each application and/or within the hub itself. While this model appears to be ideal, it rarely reflects the nature of the relationships between the associated applications. Many business processes don't easily fit into the model illustrated in Figure 1. In many enterprises business processes can (and do) cross both functional and organizational boundaries, resulting in cross-system dependencies that may exist outside the hub/spoke architecture. The data required to use these overlapping processes may be dependent on the sequence and/or frequency of the ETL processes. Integrating this data can be extremely difficult to manage.

One possible solution to this issue is to ensure that the data stored at the hub uses an open, extensible format that can be quickly and easily accessed by multiple business processes. Leveraging XML for the EAI hub helps ensure that the data is stored in a well-defined (assuming common storage schemas are used), highly accessible format.

XML and Middleware

This article has focused on the potential issues associated with EAI and how XML can be leveraged to minimize these issues. Middleware is the enabling technology for EAI. Implementing an EAI solution is simply not possible without using some sort of middleware.

The remainder of this article discusses how XML can be used to extend existing middleware tools and processes within

and across multiple enterprises. It's important to note that, while XML is capable of enabling a great many things, it can't and won't (at this stage) completely replace tried-and-true middleware solutions. It can, however, extend the flexibility and openness of these solutions.

XML and Distributed Objects

The category of distributed objects covers a fairly broad spectrum of middleware solutions. We'll categorize the following ones as distributed objects:

- Remote Procedure Calls (RPCs): This approach, while technically not objects in the classic sense, encompasses technologies such as DCE (Distributed Computing Environment).
- Common Object Request Broker Architecture (CORBA) (as defined by OMG): Enables applications from different locations and vendors to communicate with one another via an ORB (Object Request Broker) by trapping method calls and mapping them to the appropriate object.
- Component Object Model (COM)/Distributed Component Object Model (DCOM): DCOM was developed by Microsoft and is conceptually similar to DCE. Applications developed for the Microsoft environment contain interfaces that enable client objects to request services from one or more server objects regardless of their location on the network. DCOM grew out of Microsoft's COM, which enabled client and server processes to communicate on a single machine.
- Enterprise JavaBeans (EJB): EJBs are similar in concept to CORBA but are tied to a single programming language implementation (Java). The EJB architecture enables developers to create robust server-side components (written in Java) that can be reused by clients or other server-side processes.

While XML shouldn't be viewed as a replacement for the above-listed solutions, it can be used to extend their flexibility. Listed below are several initiatives currently underway to use XML with distributed objects.

RPC

Transportation of RPC messages has traditionally required the use of "closed" wire protocols such as DCOM or CORBA IIOP. The XML-RPC specification explains how proxies and stubs can communicate via HTTP and use XML as the encoding mechanism (instead of DCE RPC Protocol Data Units [PDUs]), and it's been implemented for several platforms and programming languages.



FIGURE 1 Typical EAI architecture

CORBA and COM/DCOM

The massive growth of Web-based application development prompted the W3C in May 1998 to issue a Note on incorporating the benefits of distributed objects in a platform-neutral manner. The traditional approach of POSTing HTML forms lacks the security, scalability and object interoperability of a typical distributed system. The W3C Note explains how proxies and skeletons can communicate via HTTP, XML and URIs. This approach results in a far simpler implementation since the transport mechanism (HTTP) is already provided. Formatting messages with XML enables an open and flexible standard to be used for marshaling and communicating object metadata for a CORBA interface repository or Microsoft COM/DCOM registries.

EJBs

Sun Microsystems' EJB specification is a cross-platform component architecture for the development and deployment of multitier, distributed, scalable, object-oriented Java applications. Unlike CORBA and COM/DCOM, EJB-based distributed objects are language specific (they can be written only in Java). An EJB-enabled server manages containers that in turn manage one or more EJB classes (or instantiations of classes). The container can be thought of as a "wrapper" that provides additional services (transaction control, lifecycle management and security) to the bean. The EJB deployer installs EJB classes on the server and is responsible for configuring such properties as the location, schema and transactional capabilities of the underlying database.

Sun is planning to use XML for use with EJB deployment options (also known as *deployment descriptors*). The EJB 1.0 specification relied on a serialized format that was

SELECTED EAI VENDORS

A	ATB, Inc. Active Software Adventive Technologies Affinity Media, Inc. Alier Alodar Systems, Inc. Altis, Inc. Availability, Inc.	H	HiServ Australia Pty Ltd Hitachi Computer Products (America), Inc.	P	Primus Progress Software
B	BEA Systems, Inc. Bluestone Consulting, Inc. Bluestone Software, Inc. Braid	I	IBM ISG International Software Group InConcert Information Builders, Inc. Integrated Design, Inc. Integrion International Middleware Association (IMWA) iWork Software	Q	QAD Red-Oak Software Relativity Technologies Renaissance Worldwide
C	Cambridge Information Technology Ltd (CITL) Candle Cel Corporation Clarkston-Potomac Compaq Computer Network Technology Corp. Conextions, Inc. Constellar Corporation CrossTier CrossWorlds Software	L	Level 8 Systems, Inc. Lingwood Software Liquid Software, Inc. Logica Lorac, Inc.	S	S2 Systems SAA Consultants Ltd SAGA Stellar Software Corporation Scribe Software Corporation Selectica Sequoia SilverStream SmartDB Corporation Software Associates Software Technologies Corporation (STC) Stellar Software Corporation SuperNova, Inc. Systems & Computer Technology (SCT)
D	Darc Development Corporation Data Dimensions DataBase Consulting Group, Inc.	M	M/Ware MINT Communication Systems Inc. MITEM Mastech Masterchart MicroTempus Microsoft Missat Consulting	T	TIBCO TSI Software Inc TanitObjects Tempest Software Template Software Tivoli TopTier Software
E	eFORCE Enterprise Automation Advisors, Inc. Evolutionary Technologies International (ETI)	N	N-Able Group International NEON Systems, Inc. Net Dynamics Netik New Era Of Networks (NEON)	V	VIESystems Vertex Industries Vitria Technology
F	FRONTEC AMT Forte Fujitsu FusionWare	O	ObTech Oberon Software OnDisplay Open Applications Group Oracle Osprey Systems, Inc.	W	Whittman-Hart, Inc. Wilco International
G	GE Information Services Genesis Development Corporation Glue Technology	P	PIDAS Ltd. PRL Scotland Ltd.	X	Xing

XML's self-documenting design and ability to store and communicate metadata help preserve a flexible, open-applications architecture

difficult to configure as it was hard to determine who was the information provider and who was the information consumer. As with the previously mentioned distributed object categories, messages (in this case, serialized Java objects) can be formatted with XML to maximize flexibility.

Additional information regarding the EJB specification appears at the end of this article.

XML AND MESSAGE QUEUING

The primary products in the Message Oriented Middleware (MOM) arena are IBM's MQSeries and Microsoft's MSMQ.

In June 1999 IBM announced native support for XML within its MQSeries family of products. MQSeries messages can now be formatted into XML and transmitted using the MQSeries transport. An additional MQSeries product, the MQSeries Integrator, provides the ability to bridge between XML-based and non-XML data, thereby accelerating the adoption of XML as a standard messaging format. (It should be noted that messages that are formatted using XML will result in dramatically larger message sizes, potentially impacting message processing performance.) IBM plans to provide further MQSeries support for XML via the Common Messaging Interface (CMI), which provides a

SELECTED EAI PRODUCTS


A	AMTrix	Active Works	ActiveEnterprise	Application Adapters	Electric Works Integration Framework	Evolve	Espresso Series	NEONet	Network Express Release 8.0
B	BEA eLink	BusinessBus	BusinessWare Analyzer	BusinessWare Automator	BusinessWare Communicator	BusinessWare Connectors	BusinessWare Server	OM3	Process Integrator
C	CellWare	Certified LiquidLINK Consulting	Cloverleaf	Cloverleaf Finance	Constellar Hub	Constellar WarehouseBuilder		REIMS E-business Open Transaction Server	REIMS Extranet Server
COPERNICUS								ROMA Business Service Platform	
D	DETAIL DB-Mover	DETAIL Navigator	Data Integrator	DataGate Enterprise Applications	Integration Product Suite	DataGateWay	dcServ	SMDX Gateway	STP Explorer for Workflow
E								SagaVista	Sapphire/Web
F								Shadow Direct	Skyscraper
G								TGEN for MQSeries	TPBroker
H								X-Gen	XML-Server
I	I/O Exchange	ISG Navigator	InterPlay	Interchange2000(2K)				IXpress	
J									
K									
L	LiquidLINK Plus	LiquidLINK							
M	MINT Rule Manager	MQ-IQ	MQSeries Client for Stratus VOS	MQSeries Integrator	MQexpress	Mercator			

logical message construction API similar to the MQI (Message Queue Interface, the MQSeries API). CMI will provide the ability to construct and parse messages regardless of their physical representation and will be capable of parsing both XML and language-dependent structures (such as those found in C, COBOL and Java).

While Microsoft's MSMQ product has yet to provide direct support for XML, messages can still be formatted in XML and transmitted using the MSMQ transport.

Summary

This article has attempted to document possible uses of XML within EAI and B2B integration scenarios, and a common thread running through it is formatting messages using XML. This helps avoid a tightly coupled design that can potentially lock out other applications or companies.

While XML won't immediately replace standard EAI tools and middleware solutions, its self-documenting design and ability to store and communicate metadata help preserve a flexible, open-applications architecture. 

AUTHOR BIO

John Evdemon is an architect with XMLSolutions, a software and professional services organization. John, who specializes in large-scale B2B systems using XML, has more than 11 years of experience designing, developing and implementing systems.

JEVDEMON@ACM.ORG

Tango Developer's Journal

www.tangojournal.com

Plan to attend the only **XML event** coming to

Join 1,500 XML enthusiasts, the industry's most respected technical experts, sought-after gurus and advanced users. Take advantage of 4 days of XML intellect and master new skills from those who are defining XML's future.

XML DevCon 2000

CONFERENCE:

June 25–28, 2000

EXHIBITION:


June 26–27, 2000

New York Hilton
New York City, New York

WWW.XMLDEVCON2000.COM

Presented by:  **SYS-CON
MEDIA**

 **XML
JOURNAL**

Produced by:  **CAMELOT**

Cosponsored by:  **OASIS**

New York's Silicon Alley!



Every delegate receives a FREE one year subscription to *XML-Journal* and *Java Developer's Journal* – A \$99⁰⁰ VALUE

XML-JOURNAL SETS THE STANDARD

This eagerly anticipated and widely coveted magazine, packed with information written by leading XML gurus, is the only resource you need to understand and make the most of the latest XML advancements.



BENEFIT BY ATTENDING

- The tips and techniques you'll learn will help you do your job better.
- Discover the applications being developed today that you'll need tomorrow.
- Sessions are designed for users at all levels, with special sessions just for gurus.
- Network with fellow software developers as well as recognized XML experts.
- Learn how XML is being used for large-scale enterprise applications.

TWO-DAY EXHIBIT

Free Special Events Open to All!

Exhibit Hours: Monday, June 26, 12:00 – 6:30

Tuesday, June 27, 12:00 – 6:00

The full-scale exhibit hall packed with leading vendors will be on hand to demonstrate the latest products and answer your questions.

- Keynote Presentations
- Product Education Sessions
- Technology Briefings
- Panel Discussions
- Birds of a Feather Sessions
- Product Giveaways

Register online for your **FREE Exhibit Pass – a \$25⁰⁰ value!**

TECHNICAL CO-CHAIRS

Ken North, President

KEN NORTH COMPUTING, LLC

Ken North is an author, consultant and company founder. He teaches Expert Series seminars and is writing XML, Java and Database Magic. Ken North has been a columnist for several industry trade publications and is coauthor of *Database Magic and Windows Multi-DBMS Programming*.

Max Dolgicer, Technical Director

INTERNATIONAL SYSTEMS GROUP, INC. (ISG)

Max Dolgicer is a leading authority in Enterprise Application Integration, Web Application Integration and Middleware. ISG is a New York-based consulting firm that specializes in design, development and integration of large-scale distributed applications using leading edge Internet and middleware technologies.

Max Dolgicer is a contributing editor for *Application Development Trends* magazine and a frequent contributor to a number of major trade journals.

TOPICS COVERED

The industry's recognized XML experts led by technical cochairs, Ken North and Max Dolgicer, have designed the technical program which includes 70+ sessions with over 70 hours of instruction. Topics include:

- *Leveraging XML in ColdFusion using WDDX*
- *Adding XML Capabilities with Cocoon*
- *Using Apache as HTTP Front End for an XML Database*
- *High-Performance Dynamic Web Serving Using Templates and XML*
- *Java, XML and DOM*
- *Building Enterprise Integration Servers with Message-Oriented Middleware and XML*
- *XML & Middleware: Moving Beyond Documents*
- *Simple Workflow Access Protocol*
- *The Role of XML in Knowledge Management*
- *Components, XML and DHTML: The Perfect User Interface*
- *Python & XML*
- *Hybrid Clients: The Next Generation of UI*
- *Distributed Object Architecture with XML and CORBA*
- *Automating Capital Market Activities*
- *Microsoft DAV and Its Use in Exchange*
- *XML & XSL by Xample*
- *XML and Windows DNA*
- *What's Wrong with DTDs and What's Being Done About It*

WWW.XMLDEVCON2000.COM

Media Sponsors:



XML FEATURE

WIRELESS MARKUP LANGUAGE

Using WML
to support the
presentation
layer of your
application's
architecture

The ability to access Web content and send e-mail messages from a wireless phone has become a requirement for the rapidly growing segment of wireless Internet users. Application development for wireless devices can be more challenging than conventional Web development due to the inherent characteristics of wireless devices – small display areas, narrow bandwidth connections, limited means for user input and limited memory resources. In addition, there are disparate vendor-specific wireless devices and presentation mechanisms.

To address some of these issues, the Wireless Application Protocol (WAP) Forum has defined a standard for the presentation and delivery of information to, for example, wireless phones, pagers and personal digital assistants (PDAs). The Wireless Markup Language (WML) is a specification of the WAP Forum pertaining to the presentation of content and user interface behavior on wireless devices. WML was designed to accommodate presentation on limited-capacity devices such as wireless phones.

WML is an XML language that inherits the XML document character set and most of the XML syntax. The document character set for WML is the Universal Character Set that is currently identical to Unicode 2.0. WML is specified as an XML document type definition (DTD), and is supported on any device that is WAP compliant. This article focuses on using WML for delivering content for presentation on WAP-compliant wireless devices.

Before I discuss WML, I think an overview of the WAP development model will be helpful. WAP programming is based on the Web programming paradigm. The model, as shown in Figure 1, comprises three major components.

The *client* component of the model is a WAP user agent that can interpret WML and present WML content to a user in a form appropriate for that type of wireless device. As its name implies, a *WAP gateway* converts to and from the wire-based and wireless domain, translating requests and responses from the TCP/IP protocol stack to the WAP protocol stack and vice-versa. It translates Web content into compact, encoded formats to increase transmission efficiency over the wireless network. WML can be encoded using a scheme based on the WAP Binary XML Content Format. A WAP gateway is required to access WML content over the Internet using HTTP requests. The *server* component of the model consists of a Web server, servlets and/or CGI scripts that generate WML content in response to requests.

Now that you're familiar with how WML fits into the overall WAP environment, we can begin to discuss the WML syntax.

WML Essentials

WML content is sent to a WAP-compliant device in the form of a deck. WML decks can be stored in the form of static WML files on a Web server or can be dynamically generated by CGI scripts or servlets. A WML document or deck consists of one or more cards, each of which represents a unit of interaction with a user. Typically, a card presents information and allows a user to enter text or select from a list of menu choices. A user navigates from one card in a deck to the next.

In WML, an element describes markup information about a deck. An element may be of two forms: `<tag>content</tag>` or `</tag>`. Table 1 summarizes some typically used WML elements.

Most WML elements have attributes that let you specify additional information about how a WAP device should interpret an element. All attribute values must be enclosed within single (') or double (") quotation marks. Character entities can be included in attribute values. WML supports both named character entities and numeric character entities. WML entities represent specific characters in the document character set. For example, the named entity `&` is used to represent the ampersand character (&). Note that, like XML and unlike HTML, WML is case sensitive.

A valid deck is a valid XML document; thus a valid deck must be composed of an XML declaration and a document-type declaration. The high-level syntax for defining a WML deck is shown below. All cards in a deck are defined within the `<wml>` element. Note the use of a comment (`<!-- a WML comment -->`) in the WML code.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card>
    <!-- Elements of card -->
  </card>
  <card>
    <!-- Elements of card -->
  </card>
</wml>
```

“The ability to access Web content and send e-mail messages from a wireless phone has become a requirement for the rapidly growing segment of wireless Internet users”

Examples

Given the familiarity of e-mail clients, the examples used to illustrate WML in this discussion will focus on three major use cases: (1) logging in to an e-mail server, (2) presenting a list of e-mail client menu options, and (3) sending an e-mail message.

Since the focus of this article is WML, only WML code will be used to develop the presentation component (View) of an e-mail client application for mobile devices. The Model and Controller components of this type of application (e.g., servlets, Enterprise JavaBeans) won't be discussed. However, at the end of this article there is a reference to a recent article on JavaMail that discusses how e-mail client functionality can be supported.

The code shown here was developed using the UPSDK, which provides a WAP development environment. The toolkit includes a phone simulator with a browser that supports WML. SDKs that support development using WML can be downloaded from the sites listed in the Reference section of this article.



FIGURE 1 The WAP development model

WML ELEMENT	DESCRIPTION
<wml>	Specifies a deck
<card>	Defines text and input elements that support interaction with user
<input>	Prompts user to enter text string or numbers. Attributes of this element can limit entry to numbers, set a maximum input length or specify default input values
<do>	Declares action that wireless device will perform when user presses key specified by type attribute. Label attribute specifies label to display for function key
<go>	Specifies URL to navigate to. The <go> element is used to specify task to perform <do> action. An attribute of this element defines HTTP submission method (i.e., get or post) to be used
<select>	Prompts user to choose one or more items from specified list. Attributes of this element specify single or multiple selection, title of element and default item selection
<option>	Specifies single choice in select statement. An attribute specifies URI to navigate to when option is selected
<p>	Defines paragraph. Attributes specify line-wrapping modes and alignment
	Includes image in card. Attributes specify URI for image, alternative text to display if images aren't supported, image alignment
<postfield>	Specifies name-value pair to be transmitted to server during URI request. Attributes specify field name, field value

TABLE 1 Important WML elements

Login to E-Mail Server

A core use case for e-mail clients involves a user requirement to enter a login name and password for authentication purposes. This use case requires masking the password entered by a user and sending the values back to the server (Web server, servlets) for processing.

A number of WML elements are needed to support this use case. The <input> element has a name attribute that specifies the variable that a device uses to hold user input. This element also has a type attribute that masks the password (replaces the characters by asterisks) being input. The href attribute of the <go> element is used to specify navigation to a servlet for processing the user login while the method attribute of the <go> element specifies the HTTP submission method (post). The <postfield> element passes the login name and password to the HTTP server receiving the <go> request. The value attribute of the postfield element in the code below uses WML variables. For example, the value of the WML variable \$pwd is substituted at runtime with the actual password entered by the user. A code snippet for entering the user ID and password is given below. A more complete code listing is shown in Listing 1.

```
<do type="accept">
  <go method="post" href="http://localhost:8080/servlet/LoginServlet">
    <postfield name="user" value="$userid&"/>
    <postfield name="password" value="$pwd&"/>
  </go>
</do>
<p>
  Password:
  <input name="pwd" type="password"/>
</p>
```

A simulated wireless device presentation and navigation of the two WML cards defined in Listing 1 that allows a user to log in to an e-mail system is displayed in Figure 2.

Main Menu of E-Mail Client

After logging in to an e-mail server, a user is typically presented with a list of menu options (e.g., send an e-mail message).

A number of WML elements and attributes are needed to support this use case. The <select> element is used to present the user with e-mail menu choices. The onpick attribute of the <option> element specifies the card to navigate to when that menu option is selected. In this example, if the user wants to send a message, the next card shown is the card within the deck with the "send" ID. The id attribute of the <card> element specifies a name that lets you navigate to the card from other cards in the same deck. The code below shows how e-mail menu options may be presented to a user.

```
<!-- Sample card for showing main e-mail menu -->
```

```
<card id="main">
  <p mode="wrap">
    My E-mail Messenger
  <select>
    <option onpick="#send">Send a message</option>
    <option onpick="#get">Get messages from server</option>
    <option onpick="#logout">Logout</option>
  </select>
  </p>
</card>
```

A display of the presentation of this WML card, showing a user e-mail menu choices, is shown in Figure 3.

Sending an E-Mail Message

From the main menu an e-mail user can create and send a simple text e-mail message. The "send" card is navigated from the main menu, as shown in the previous example. Multiple cards are needed to accomplish this use case.

The WAP phone simulator used for development has a programmable soft key. The <do> element is used to map an action to a soft key. The href attribute of the <go> element is used to specify navigation to the next card to be displayed. A code snippet from the first card used to send an e-mail message is shown below. A more complete code listing for sending an e-mail message is shown in Listing 2.

```
<do type="accept" label="ok">
  <go href="#subject"/>
</do>
<p>
  To:
  <input name="to"/>
</p>
```

A display of the presentation and navigational sequence of the three WML cards defined in Listing 2 for creating and sending an e-mail message is shown in Figure 4.

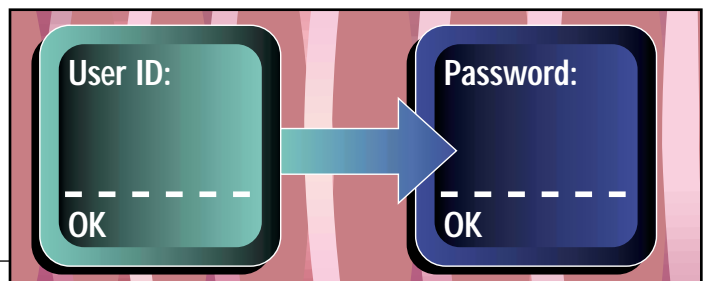


FIGURE 2 WAP client display of user login

Sequoia Software

www.xmlindex.com



FIGURE 3 WAP client display of main e-mail menu

WAP client validation of user input is an important consideration, given the inherent limitations of wireless devices. WMLScript could be used to validate user input such as an e-mail address before it's sent to a servlet for further processing. For those who have used HTML scripts, WMLScript is a procedural scripting language based on ECMAScript and adapted to suit the needs of wireless devices.

Conclusion

If you're developing applications for wireless devices, you should consider using WML to support the presentation layer of your application's architecture. WML provides a standard and extensible delivery mechanism of content for presentation on a plethora of display-constrained, WAP-compliant mobile

devices. WML provides a developer with a means to focus on the content to be displayed rather than on the variety of proprietary presentation mechanisms of mobile devices. For more information on WML and SDKs with support for WML, please see the references below. 🌐

References

1. Wireless Markup Language Specification Version 1.1: www.wapforum.org
2. UPSDK WAP Software Development Kit: www.phone.com
3. Nokia WAP Toolkit: www.forum.nokia.com/
4. Moraes, I. (1999). "JavaMail: Framework for Developing Internet-Based E-Mail Client Applications." *Java Developer's Journal*, Vol. 4, issue 10.
5. WMLScript Language Specification: www.wapforum.org

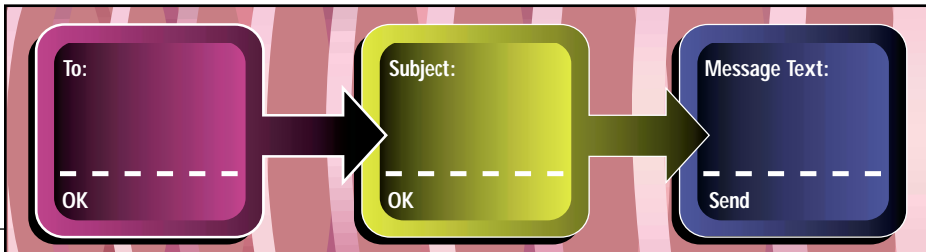


FIGURE 4 WAP client display of cards for sending e-mail

AUTHOR BIO

Ian Moraes, Ph.D., a principal engineer at Glenayre Electronics, has developed client/server systems for both the telecommunications and financial services industries. Currently he works on the architecture, design and development of a unified messaging application.

IMORAES@ATLANTA.GLENAYRE.COM

LISTING 1

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<!-- LISTING 1 -->
<!-- Two cards used to accept userid and password -->
<!-- First card gets user id and then goes to password card -->
<wml>
<card id="userIdLogin">
  <do type="accept">
    <go href="#pwdLogin"/>
  </do>
  <p>
    User Id:
    <input name="userid"/>
  </p>
</card>

<!-- This card accepts password and sends values -->
<!-- to Login servlet for authentication -->
<card id="pwdLogin">
  <do type="accept">
    <go method="post"
href="http://localhost:8080/servlet/LoginServlet">
      <postfield name="user" value="$userid&amp;"/>
      <postfield name="password" value="$pwd&amp;"/>
    </go>
  </do>
  <p>
    Password:
    <input name="pwd" type="password"/>
  </p>
</card>
</wml>
```

LISTING 2

```
<!-- Three cards used for sending an e-mail message -->
```

```
<!-- Note this is not a deck -->

<!-- First card is for entering recipient of e-mail -->
<!-- then navigate to next card for subject of message -->
<card id="send">
  <do type="accept" label = "ok">
    <go href="#subject"/>
  </do>
  <p>
    To:
    <input name="to"/>
  </p>
</card>

<!-- This card is for entering subject of message -->
<card id="subject">
  <do type="accept">
    <go href="#msgtxt"/>
  </do>
  <p>
    Subject:
    <input name="subject"/>
  </p>
</card>

<!-- This card accepts message text and sends values -->
<!-- to Send E-mail servlet for transmission to E-mailServer-->
<card id="msgtxt">
  <do type="accept" label="Send">
    <go method="post"
href="http://localhost:8080/servlet/SendServlet">
      <postfield name="to" value="$to&amp;"/>
      <postfield name="subject" value="$subject&amp;"/>
      <postfield name="text" value="$text&amp;"/>
    </go>
  </do>
  <p>
    Message Text:
    <input name="text"/>
  </p>
</card>
```



Software AG

www.softwareag.com/tamino



How XML industry specifications are being developed

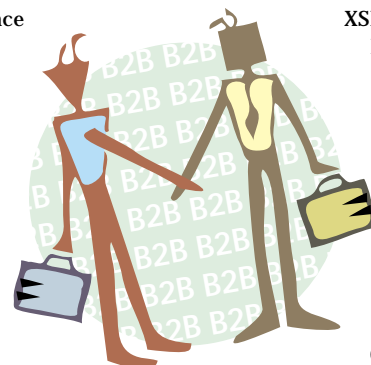
OASIS and XML.org

XML is a significant technological achievement, but what's it really good for when it comes to e-business and industry applications? In these columns I'll discuss how companies and consortiums are developing XML specifications for a wide range of industries. Most of the hot activity these days is around using XML for messaging for business integration and business-to-consumer (B2C) and business-to-business (B2B) e-commerce. I don't expect to discuss publishing in any depth, though some interesting work is going on with content distribution such as ICE and newspaper electronic formats such as XML News, and I'll probably talk about it in future columns.

In this inaugural column I'll discuss OASIS and XML.org and their role in helping XML industry specifications get developed and used. I'm on the board of directors of OASIS and am a member of the XML.org steering committee helping represent IBM. The opinions expressed below are my own and don't necessarily express those of IBM or other members of the OASIS board or membership.

OASIS is the Organization for the Advancement of Structured Information Standards (www.oasis-open.org) and is a nonprofit international consortium. OASIS, previously known as SGML Open, was founded in 1993, and as the older name implies, much of its early activity reflected SGML's strong presence in the document and publishing community. With the ascendancy of XML and its importance for business message formats, the name change became important. OASIS now has more than 100 member organizations, and recent membership gains reflect this shift in emphasis from documents to transactions. We recently made a strong commitment to our European members by hiring a representative based in the UK. Membership growth has accelerated in recent months and we expect to hire representatives in other parts of the world, such as Asia Pacific, as suitable candidates become available.

The "structured information standards" in the expansion of the OASIS acronym currently refers to four technologies: XML, SGML, HTM and CGM. CGM, the Computer Graphics Metafile, is a format defined by ISO /IEC 8632:1992 for describing vector, raster and hybrid (raster and vector) graphics in a compact way. The CGM Open Consortium (www.cgmopen.org) is an affiliate organization of OASIS and consists of vendors and users of CGM technology. Since CGM isn't based on XML, I'm not going to discuss it in detail here. It's likely that OASIS will soon have other affiliates that are dedicated to using XML for particular industries. The advantage of being an OASIS affiliate is that you and your colleagues can concentrate more on the domain-specific work at hand rather than getting overly involved with consortium administration factors. An affiliate does have its own board of directors and bylaws, however, and thus lies somewhere between a completely independent consortium and a technical committee.



One of the real jewels of OASIS is the Robin Cover SGML/XML pages. These pages include more than 3,000 documents that discuss the technology that's been developed for SGML and XML over the last several years. Within these pages you'll also find descriptions of much of the industry activity involving XML. Robin lists the essential books to read and has descriptions of public and commercial software for XML, SGML and related technology such as XSL.

One of the most exciting things to happen within OASIS recently was the formation of XML.org (www.xml.org), an initiative to create a vendor-neutral clearinghouse for XML resources. Typically these resources are DTDs or schemas, but they might be XSL stylesheets or HTML pages. Nine OASIS member organizations were the initial sponsors of XML.org. IBM, Oracle, SAP and Sun Microsystems each contributed \$100,000 and Commerce One, DataChannel, Documentum, GCA and SoftQuad each contributed \$25,000. The contribution level is based on corporate revenues, not dedication to the initiative!

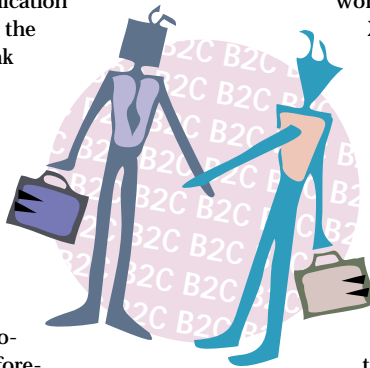
Right now, it's very hard to find detailed technical XML specifications and information about work being done in industry via a Web search. It's important to have an XML resource clearinghouse where

you can go to access the latest DTDs and schemas and learn how they should be used. The clearinghouse must be available to you from your Web browser, but also for application software that uses the XML resources (think of a program that needs the latest version of an XSL stylesheet for a transformation).

The physical clearinghouse will be built using the OASIS/XML.org registry and repository architecture. We foresee a global Web of repositories linked via a master registry of the contents and metadata. So XML.org will have a repository of schemas and other resources, but there will be compatible repositories out there that are under the control of independent organizations. These organizations might want their own repositories so they can control updates, versioning or access control. Their repositories will be compatible with that used by XML.org itself, because they all adhere to the API defined within the OASIS/XML.org registry and repository specification. While there may be open implementations of the repository, we also expect commercial versions built on top of enterprise-quality databases from vendors such as IBM, Oracle and others.

You'll be able to query the master XML.org registry on the contents of all the repositories within the Web. The repository at the XML.org site will exist for the convenience of those who don't wish to maintain their own installation. The XML.org repository will also hold the work done by OASIS technical committees and this leads to an important, sometimes misunderstood, point about the relationship between OASIS and XML.org.

XML.org is an activity within OASIS; it's not a separate organization. In some sense it's a branding for much of the XML activity that takes place within OASIS. All "XML.org technical committees" are actually "OASIS technical committees," and representatives of OASIS organizations can participate in any of the technical work done under the XML.org banner. So while we say that XML.org has the nine original sponsors, it's actually a resource and a benefit for all 100+ OASIS member organizations. The registry and repository technical committee, for example, had existed for several months within OASIS before



XML.org was announced. The XML.org clearinghouse will be the result of the technical work of many OASIS members. It'll also be influenced by the work done within the XML/EDI community (see www.xmledi.org), the Object Management Group (OMG) and others. Incidentally, OASIS and the OMG (www.omg.org) have exchanged memberships so they can more easily take part in, and advantage of, each other's technical work.

Part of the mission of OASIS is to be an organization in which member companies can come together and create XML industry standards. In addition to the affiliates I mentioned above, technical work can be done within technical

this. From the XML.org site you'll first get access to the XML resource registry and repository. Since this isn't in place yet, the site has a catalog of XML DTDs and schemas. Where there's a description of a specification within Robin Cover's pages, the catalog includes a link. We update the catalog as new work is brought to our attention, and the site has a form for submitting work you wish to have listed. The catalog will be the basis for populating the XML.org registry and repository when it goes online early this year.

Via the Robin Cover pages and other links, the XML.org site is a source of timely information about the application of XML in industrial settings. This includes Robin's daily news briefs about XML, which are syndicated to XML.com and other sites. The OASIS news page provides an update on XML activities at OASIS and its member companies. To my knowledge, the XML.org calendar is the most complete catalog of XML industry events.

“ Part of the mission of OASIS is to be an organization in which member companies can come together and create XML industry standards ”

committees. In a process somewhat similar to that in the W3C, interested parties produce a briefing package of what they hope to accomplish in a specified timeframe. Once they get approval from the OASIS board to proceed, the technical work begins, perhaps with input from work already done within member companies. The goal is to eventually reach the status of XML.org Recommendation. We're investigating a process by which we can also award this status to high-quality XML specification work done by other consortiums. The tricky and perhaps controversial part is deciding on exactly what *high quality* means! One thing that's certain is that an XML.org Recommendation schema must be written using a standard description language such as the XML DTD format or the forthcoming W3C XML Schema language. Compliance to real, open standards rather than a single company's XML vision is at the core of XML.org, and that starts with how we'll describe our recommended specifications.

When you visit the XML.org site, you'll see that it's described as the "XML Industry Portal." While I know that *portal* sometimes gets overused these days, let me explain exactly what we mean by

OASIS has just hired a managing editor for the XML.org site and we expect that this will lead to even greater coverage of industry activities. We plan to have industry-by-industry descriptions of the major development work on XML specifications. For example, the financial industry page will discuss the significance and status of the FpML, FIXML and the S.W.I.F.T. XML specifications. We also plan to have technical articles about XML specifications written by people in the industries who are actually doing the development. If you're interested in writing such an article or have specific topics that you wish me to discuss in future columns, please contact me via the e-mail address listed below.

In my next column I'll discuss the work taking place within the joint UN/CEFACT and OASIS Electronic Business XML initiative (www.ebxml.org). After that, I'll look at standards work related to EDI and XML, and start investigating XML activities in specific industries. ☛

AUTHOR BIO

Bob Sutor is a member of IBM's XML Strategy and Technology Team and is chief strategy officer of OASIS. While a member of IBM's research staff, Bob developed applications and led advanced technology projects related to XML and Internet publishing.

SUTOR@SUTOR.COM

XML FEATURE

PRESENTING XML TO THE WEB

One of many
approaches to
developing
Web-based
applications

Every problem in software development can be approached in multiple ways. Graphical user interface (GUI) development is no exception to this rule. With increasing numbers of Web-based and e-commerce applications around the globe, the greatest demand so far – which is growing every day – has been for thin clients or Web clients. The question is whether XML can help in this area. The answer is yes, but only if it satisfies your solution requirements.

In this article we'll see how XML can be used to develop Web-based user interfaces. XML may present an elegant approach to Web UI development, but it has to overcome the challenge of being able to work in today's Web browsers and be cross-browser compatible. This is the article to read before you invest in XML for developing Web clients.

What Is a Web UI?

Generally speaking, a Web UI means a program that can be executed within any entity that acts as a gateway to the WWW or the Internet. The most common Internet gateways so far have been the Web browsers such as Internet Explorer (IE) or Netscape Communicator. Because of Internet bandwidth limitations and security constraints, most Web UIs consist mainly of HTML pages. Anyone who has written HTML code or scripts knows how difficult it can be to manage and maintain the code. Moreover, it's a real nightmare to support and maintain multiple versions of your HTML pages when customers ask for a different look and feel to your presentations. Thanks to the latest Web technologies, such as JavaServer Pages (JSP), Servlets and XML, you can provide a simple, robust, scalable, maintainable and reusable programming model to generate static and dynamic contents for Web browsers.

Figure 1 shows an example of a Web-based system that generates dynamic HTML. JSPs are used to receive HTTP requests from Web clients. (The detailed architecture and functionality provided by JSPs is beyond the scope of this article.) In brief, JSP is an extension of Servlet technology that provides an easy way to build reusable, component-based Web pages with dynamically generated platform-independent content. JSP pages are dynamically compiled into Java Servlets that actually act as a Controller interacting with the business logic or data transformation layer. The business logic layer can be implemented using Enterprise JavaBeans (EJB). The data that is obtained from business logic in XML form can be converted to HTML form using XML transformation (XT) tools, as shown in Figure 2. The XT component applies XSL (Extensible Stylesheet Language) business data to generate the HTML. The following section describes XML, XSL and XT in more detail.

XML-BASED UI?

How exactly does XML help in developing Web UIs? The answer to that question lies in how XSL and XT actually work. XSL is used to specify rules for transforming input XML files into HTML. Note that XSL can also be used instead of HTML to convert from one XML format to another. Before you begin defining rules, you must know the input XML and the output HTML. Based on the output requirements, you can start defining templates or rules using the XSL specification. These rules define how to process tags within XML documents. Once you've finished defining your XSL rules, you can apply those rules to the XML document using one of the XSL transformation (XT) tools. Several XT transformation tools or utilities are available in the market (see references at the end for hyperlinks).

Figure 2 shows how XT generates HTML from XML and XSL inputs.

Typically, you can apply XT on your XML/XSL documents from the command line or through JavaBeans or Java Applets. In the following section we'll see how to apply XT from the command line as well as in a program to generate an HTML page.

Why Use XML in UI?

There are a number of reasons for application developers to be motivated to use XML in their applications and for generating Web contents. Owing to its self-describing and semistructured nature, XML can be widely used to describe the metacontent of documents. It can be used for publishing and exchanging database contents, and as a messaging format for communication between application programs. XML and XSL together provide a clean separation between presentation and data. This means you can have various XSL for the same XML data to produce various HTML pages, each with a potentially different look and feel. This can be useful for personalization and customization of Web applications. You can also use XSL efficiently to set up different kinds of filters on XML data to display or process only a subset of the data. Use of XSL doesn't require any special graphic components such as applets or ActiveX controls. This helps to keep your Web UI lightweight and pure HTML.

In addition, XML documents can handle international character sets. The XML 1.0 Recommendation is defined based on the Unicode character set, which is crucial for developing global cross-enterprise applications in today's world. In the future, you may not even have to apply the XSLT transforms manually because the Web browser will render the XML documents appropriately based on the specified XSL. In fact, IE 5.0 already supports this capability and Netscape is expected to provide this ability in their next release. Following is an example that shows partial XML data for a real estate application and its corresponding XSL for processing the XML.

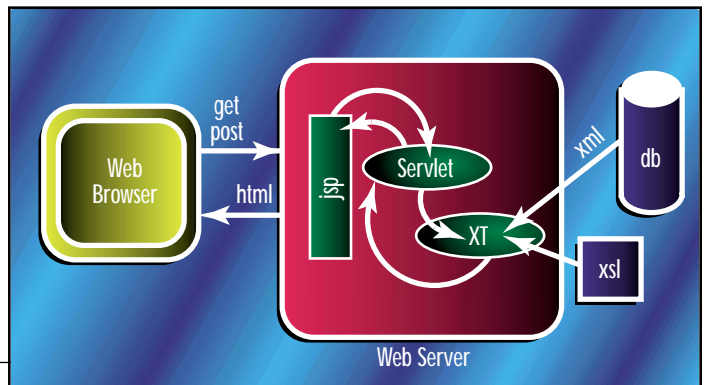


FIGURE 1 A design for a Web-based UI

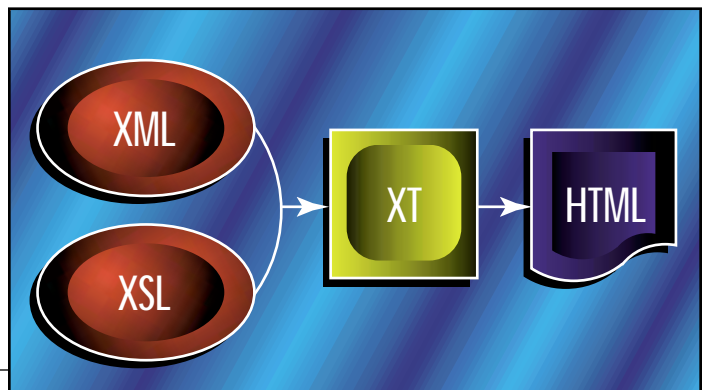


FIGURE 2 XML + XSL = HTML

XML TAGS FOR REALESTATE.XML

```
<HOUSE>
  <PRICE>150000</PRICE>
  <BEDROOMS>2</BEDROOMS>
</HOUSE>
```

XSL RULES FOR PROCESSING THE XML TAGS IN REALESTATE.XML

```
<!-- For each HOUSE element contained in the HOUSES
document element, process the content of the
xsl:for-each as a template. -->

<xsl:for-each select="HOUSE">
  <!-- The '.' context is now the "HOUSE" element -->
  <TR>
    <TD class="cell">
      <!-- gets the text from the "PRICE" element -->
      <xsl:value-of select="PRICE"/>
    </TD>
    <TD class="cell">
      <!-- gets the text from the "BEDROOMS" element -->
      <xsl:value-of select="BEDROOMS"/>
    </TD>
  </TR>
</xsl:for-each>
```

Why Not HTML?

You may be wondering why, if XML can be used to generate HTML, you can't write the same HTML directly yourself. The answer is that XML is much more flexible and maintainable than simple HTML. HTML is a simple markup language, with a fixed set of tags that are used only for Web documents. Moreover, most of these tags pertain to the presentation of the document on the Web. There isn't much support for extracting data from within an HTML document in a flexible fashion independent of its presentation tags. Thus the problem of limited flexibility can be solved with XML.

With XML, you can define your own set of tags by means of a DTD (document type definition). XML separates presentation and data and allows dynamic data interchange. In addition, you can easily perform data validation and represent complex data structures with rich relationships using XML. XML can be used to create much richer documents than HTML.

Web browsers are lenient when they render HTML documents on the Web. This means that even if some of the tags are ignored or handled incorrectly, the browser doesn't really complain. The information will be displayed on the screen in a reasonable fashion. The human reader will have to read it and make sense out of it. This will cause problems if you're developing complex Web applications. With XML, validating parsers can be made responsible for generating well-formed and valid documents.

Another area in which XML can be more useful than HTML is for representing metacontent or metadata. HTML has limited tags (such as TITLE and META) for accommodating metacontent. In addition, these tags can be used only within the HTML document. That means that it's not possible to search through the metacontent without downloading the whole HTML document. XML's various metacontent formats such as

“ XML adds more flexibility, reusability and maintainability to your presentation layer when compared to pure HTML ”

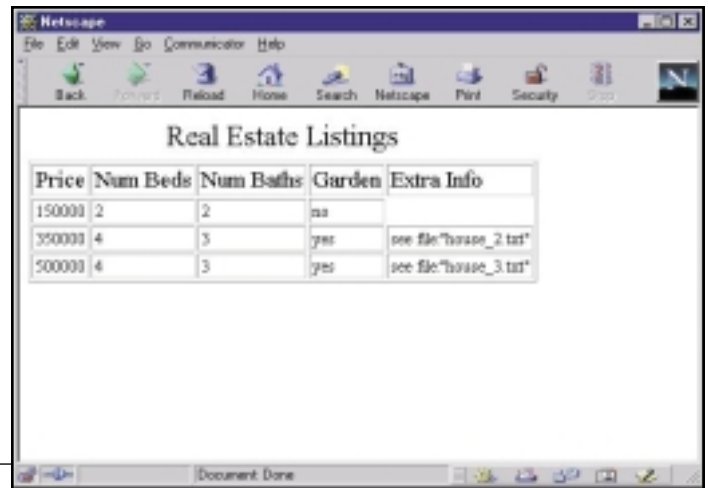


FIGURE 3 HTML generated after XSL transformation

RDF, CDF and OSD can be used to store the metacontent externally to an XML file. This provides flexibility, extensibility and performance benefits in e-commerce environments. Thus the benefits of XML far outweigh those of simple HTML. In the next section we'll work with a practical example to see how to generate HTML content by using XML and XSL.

XSL and XT

Before you begin, make sure you've downloaded the necessary classes and packages for parsing XML and XSL and applying XT. In the example James Clark's parsers and XT packages are used for transformation. Here, XML data is defined in a file called realestate.xml (see Listing 1) and XSL rules are defined in realestate.xsl (see Listing 2). Assuming you've installed the XT package in your C:\xt\ directory, one way to generate HTML would be from the command line using the following:

```
C:\xt> xt xml_file.xml xsl_file.xsl > html_file.html
```

In our example,

```
C:\xt> xt realestate.xml realestate.xsl > realestate.html
```

This would produce HTML code in realestate.html, which can be directly displayed in the browser.

Now let's see how you can achieve the same results programmatically by calling XT from within a Java Servlet to generate HTML and send the response back to the browser. In the code snippet in Listing 3 we create a new instance of an XML parser and XSL processor. Then we set the ServletOutputStream as the output stream for the XSL processor. Next we set the XSL as the stylesheet input. Finally, we feed the XML input to the parser and that's it! The result of XT, which is an HTML, is streamed as a ServletResponse back to the browser and the browser displays the same HTML as shown in Figure 3.

The code in Listing 3 is the function that transforms XML to HTML by applying XSL. This can be implemented as part of the servlet or as a bean that will be called from the servlet. To keep the demo code simple, error checking and exception-handling code is not shown in the listing.

So, What Is the Challenge?

So far we've seen how XML/XSL can provide an elegant approach to dynamically generating rich Web contents. XML adds more flexibility, reusability and maintainability to your presentation layer when compared to pure HTML. But, as with almost every solution, this solution has its own issues and challenges that can make it unsuitable for some applications. To start with, XML technology itself is immature. Standards and specifications for XML and XSL are still evolving, and change frequently. This means frequent updates to your XML document definitions, gram-

VSI

www.vsi.com/breeze

mar and XSL templates. In addition, you may be required to keep up with the rapidly changing versions of XML tools (such as the XML parser and XT). Another issue with XSL is the learning curve for someone who hasn't worked with any type of markup language. Depending on the type of XML you want to process, the XSL stylesheet can be quite complicated. For example, if you have a highly nested and hierarchical data structure that you want to represent as a complex tree, or an HTML table as a Web page, XSL rules can be quite complex. Also, as the complexity and size of the data grows, it will have an effect on the overall performance of the transformation process using XSLT.

BROWSER AND XSL ISSUES

In developing a browser-based application, the normal tendency is to assume that code that works in one browser will run under other browsers or different versions of the browser. If you don't want to be disappointed at the end, get rid of that assumption! Netscape and IE, for example, are different in the way they interpret some of the HTML and JavaScript code. The best practice would be to test your application (code) under both browsers with as many different browser versions as you need to support. If you have a wide customer base that you need to support, it would be a good idea to choose a browser type and version that you want to support based on the lowest common denominator. Different browsers have different ways of interpreting certain special characters. You may face this problem even with XSLT because certain versions of XT tools have special meaning for certain special characters such as "%." This means that if you have any of these characters in your XSL, XT may fail to generate the correct HTML code unless the characters are encoded in a special fashion.

There are some subtle issues you may have to tackle while writing XSL. For example, syntax that's acceptable as XML may not be valid as HTML. For example, if you have the following line in your XSL file:

```
<SCRIPT language="JavaScript" src="filename.js"></SCRIPT>
```

When compiled, using XT, the generated HTML would look like this:

```
<SCRIPT language="JavaScript" src="filename.js"/>
```

This is valid XML but not valid HTML and can cause severe problems (since the script tag isn't closed, your JavaScript functions will probably not be available). Here's one solution. By adding something between the tags, the result will be correct.

```
<SCRIPT language="JavaScript" src="filename.js"><xsl:comment>some text</xsl:comment></SCRIPT>
```

The foregoing line yields valid HTML as:

```
<SCRIPT language="JavaScript" src="filename.js"><!-- some text --></SCRIPT>
```

There will be more issues and inconsistencies that you'll encounter once you begin coding for cross-browser-, cross-version-compatible applications.

Conclusion

An XML-based approach is one of many approaches to developing Web-based applications. XML provides promising and flexible technology for developing Web applications that need to integrate seamlessly in a heterogeneous environment (a typical e-commerce environment). The XML/XSL model provides a useful way of representing or publishing your enterprise data on a Web page. But you need to evaluate your requirements carefully to avoid unexpected pitfalls. Various other technologies can be used along with XML/XSL to overcome some of the deficiencies while exploiting the benefits of XML. These include JSP, applets and ColdFusion.

One thing to keep in mind is that a technology that's relatively less efficient but open and easy to understand – and that provides more flexible solutions compared with its competitors – will have a better chance to win in the Internet world. Good luck! 🍀

XML References

- XML Repository: www.xml.org
- W3C consortium: www.w3c.org
- Sun: <http://java.sun.com/xml>
- IBM: www.alphaworks.com
- OASIS: www.oasis-open.org
- Webmethods: www.webmethods.com
- Bluestone: www.bluestone.com
- Sun's ProjectX: <http://developer.java.sun.com/developer/products/xml/>

AUTHOR BIO

Bhaven Shah, a Sun-certified Java programmer, has more than five years' experience in object-oriented programming with a focus on Java for the past three years. Bhaven's expertise is in developing distributed component-based architectures and GUIs, with a current emphasis on design and development of enterprise-wide Web-centric applications. He holds BS and MS degrees in computer science.



<p>LISTING 1 REALSTATE.XML</p> <pre><?xml version="1.0"?> <!DOCTYPE houses [<!ELEMENT house (price,bedrooms,baths, garden?, extra_info?) > <!ELEMENT price (#PCDATA) > <!ELEMENT bedrooms (#PCDATA) > <!ELEMENT baths (#PCDATA) > <!ELEMENT garden (#PCDATA) > <!ELEMENT extra_info (#PCDATA) >]> <HOUSES> <HOUSE> <PRICE>150000</PRICE> <BEDROOMS>2</BEDROOMS> <BATHS>2</BATHS> <GARDEN>no</GARDEN> </HOUSE> <HOUSE></pre>	<pre><PRICE>350000</PRICE> <BEDROOMS>4</BEDROOMS> <BATHS>3</BATHS> <GARDEN>yes</GARDEN> <!-- the attachment is the additional information about this house --> <EXTRA_INFO>see file:"house_2.txt"</EXTRA_INFO> </HOUSE> <HOUSE> <PRICE>500000</PRICE> <BEDROOMS>4</BEDROOMS> <BATHS>3</BATHS> <GARDEN>yes</GARDEN> <!-- the attachment is the additional information about this house --> <EXTRA_INFO>see file:"house_3.txt"</EXTRA_INFO> </HOUSE> </HOUSES></pre>
<p>LISTING 2 REALSTATE.XSL</p> <pre><?xml version="1.0"?> <!-- Sample of styling in an imagined realstate document. --></pre>	

```

<!-- Note the XSL namespace declaration. -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0"

                xmlns="http://www.w3.org/TR/REC-html40"
                result-ns=""

<!-- The root template... processing begins here.
The root is not the document element, but the node
above the document element that can also have com-
ments, and processing instructions. -->

<xsl:template match="/">

<HTML>

<HEAD>

<META http-equiv="Content-Type" content="text/html;
charset=iso-8859-1"/>

<META http-equiv="Expires" content="0"/>

<STYLE TYPE="text/css">

<!-- Create some CSS patterns that will match the
HTML elements created below. This illustrates
that CSS can be used with XSL. -->
BODY { font-size:10pt; }

.cell { font-size:10pt; }

.listingCaptions { font-size:18pt;}
.tableHead { font:bold;}
</STYLE>
</HEAD>
<BODY>
<DIV>
<!-- Call the child of the root, which is at least
the document element. Other children of the root
can be comments, and processing instructions. -->
<xsl:apply-templates/>
</DIV>
</BODY>
</HTML>
</xsl:template>

<!-- The template for the document element. This template
could be directly embedded in the template above, but
it's nice to break large templates up a bit. -->
<xsl:template match="HOUSES">
<!-- The '.' context is now the "HOUSES" element -->
<TABLE border="1" frame="border" rules="all" cell-
padding="2">
<CAPTION class="listingCaptions">Real Estate List-
ings</CAPTION>
<COLGROUP>
<COL width="30" align="right"/>
<COL padding-left="15"/>
<COL padding-left="15"/>
<COL padding-left="15" align="center"/>
</COLGROUP>
<THEAD class="tableHead">
<TD>Price</TD>
<TD>Num Beds</TD>
<TD>Num Baths</TD>
<TD>Garden</TD>
<TD>Extra Info</TD>
</THEAD>
<!-- For each HOUSE element contained in the HOUSES
document element, process the content of the
xsl:for-each as a template. -->
<xsl:for-each select="HOUSE">
<!-- The '.' context is now the "HOUSE" element -->
<TR>
<TD class="cell">
<!-- get the text from the "PRICE" element -->
<xsl:value-of select="PRICE"/>
</TD>
<TD class="cell">
<!-- get the text from the "BEDROOMS" element -->

```

```

<xsl:value-of select="BEDROOMS"/>
</TD>
<TD class="cell">
<!-- get the text from the "BATHS" element -->
<xsl:value-of select="BATHS"/>
</TD>
<TD class="cell">
<!-- get the text from the "GARDEN" element -->
<xsl:value-of select="GARDEN"/>
</TD>
<TD class="cell">
<!-- get the text from the "EXTRA_INFO" ele-
ment -->
<xsl:value-of select="EXTRA_INFO"/>
</TD>
</TR>
</xsl:for-each>
</TABLE>
</xsl:template>

<!-- Note how much smaller and manageable this is
compared to the equivalent raw HTML. -->

</xsl:stylesheet>

```

LISTING 3 CODE SNIPPET FOR APPLYING XT FROM JAVA PROGRAM

```

@param xmlInput source of xml
@param xslInput source of Xsl
@param out destination of output in the case of Servlet, it's
the OutputStream obtained from HttpServletResponse
@return true if transform succeeds otherwise false.

public boolean transform(InputSource xmlInput, InputSource
xslInput, OutputStream out)
{
    Parser parser = null;
    String parserClass = "com.jclark.xml.sax.Driver";
    try
    {
        // create a new parser instance
        parser = (Parser)Class.forName(parserClass).newIn-
stance();
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }

    // instantiate a new XSL processor instance
    XSLProcessor xsl = new XSLProcessorImpl(parser);
    ResultTypeHandlerImpl resultTypeHandler = new
ResultTypeHandlerImpl(xsl);
    xsl.setResultTypeHandler(resultTypeHandler);

    // set ServletResponseStream as the result type handler
    resultTypeHandler.setOutputStream(out);
    xsl.setStylesheet(xslInput);

    try
    {
        xsl.parse(xmlInput);
        return true;
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
    System.out.println("parsing failed...");
    return false;
}

```





Process data in a simple and straightforward manner

XML and Messaging

XML has been used in applications as a means of passing data between heterogeneous applications, to provide meta-information over content and maintain structure in data. Simply put, if HTML is the language to display information, XML is the language that can speak business terms or jargon.

In this article I'll discuss the fact that XML enables applications to organize and process information better on the enterprise. For this application I refer to Java technology for mail, servlets and messaging.

Messaging

Messaging is an integral part of most Internet-related applications because it handles data better for easier transaction management. It's used in the enterprise to exchange data between two heterogeneous applications.

For example, data comes into the enterprise through the Internet and then is sent to the back-end ERP application/system. The messages come via HTTP and have to be processed into specific formats for the appropriate back-end system.

Data is either queued and sent to one application listening at the other end of the queue, or broadcast to multiple applications wanting the same data. The messages from the queue are read by applications that process the message and translate its content for the ERP application.

Enterprise-wide applications have a messaging layer that sits between the Web server and the application servers and facilitates the delivery of appropriate messages to the appropriate servers. When a message is delivered in a queue, the queue acts as a buffer and holds the data until the server services the request. In addition, the messaging layer can have transaction management built in that would allow for the retention of messages in case the application server crashes, thus preventing the loss of critical data for the enterprise. It also prevents the handling of duplicate messages.

What Are These Messages and What's Their Format?

The following are messages that come into the enterprise in various structures or formats:

FLAT MESSAGES

In this format the data coming into the enterprise isn't structured. There's no relation between the various elements that form the content of this type of message.

For example, data submitted through a customer information form contains various heads such as the name, address, phone number, and so forth, but there's no hierarchy in this structure. Each element qualifies itself and doesn't depend on the other for its existence.

HIERARCHICAL MESSAGES

When it comes to data in the hierarchical format, there's a distinct relationship between the various elements that make up the data.

For example, an EDI message is made up of segment groups containing segments that in turn contain data elements and finally data. Each segment has to be within a segment group and the data element has to be with the segment. These could be mandatory or optional within their ancestor, but by necessity their existence is based on the existence of the ancestor.

Why XML? What Role Does It Play?

PLATFORM INDEPENDENCE

XML is platform independent. It can be used as a medium to send data between heterogeneous applications without each application having to know about the proprietary format of the other. If I have two word processing systems that need to

transfer content between each other, they could do so without knowing about the other's format since all they need to do is structure data as XML and send it across. Data could be qualified to be, for example, para or lesson:

```
<PARA> This is a para</PARA>
<QUOTE> The early bird gets the
worm</QUOTE>
```

or more meaningfully based on the application.

```
<NAME>Abc</NAME>
<AILMENT>def</AILMENT>
<PRESCRIPTION> List of
medicines</PRESCRIPTION> for a
doctor.
```

XML for Hierarchical Messages

Since XML is a structured language, it's a perfect fit for hierarchical types of messages; as data can be easily mapped to elements, the XML document, as a tree structure, takes care of the hierarchy maintenance. With an XML parser it's easier to extract difficult data from messages such as EDI because the parser does the job of isolating the data. It's easy to figure out how many times a particular element occurs as a child of which node in the tree. For example, in the case of EDI it could be represented as:

```
<SEGMENTGROUP id=1>
  <SEGMENT mandatory="true">
    <DATA ELEMENT>abc</DATA ELEMENT>
  </SEGMENT>
  <SEGMENT mandatory="false"></SEG-
MENT>
</SEGMENTGROUP>
```

Let's consider an EDI message (see Listing 1). When an EDI message is processed, it's required to maintain the EDI-specific tags as well as user-fed data.

EDI is divided into segment groups and segments, and each of the segments as well as the segment groups is mandatory or optional. Each segment can be repeated a number of times, indicated as 99 for 100 times and 999 for 1,000 times, which means 100 or 1,000 is the maximum.

XML plays a perfect foil to maintain EDI-specific information and hold data. One way in which an EDI message could be transformed into an XML document is to convert each of the EDI segment names to element tags. The *properties* of the segment, such as whether it's mandatory or optional, can be taken care of in the DTD; a *loop counter* can mention the number of times this occurs and be placed as attributes for the element. *Identifiers* can identify each of the segments. The *EDI separators* could be placed in the element attribute list as separator attributes for each element.

Although an EDI message could carry a DTD for ease of use, one could also use standalone, well-formed documents. Since the EDI structure itself doesn't have an overlapping hierarchy of segments, a standalone document is fine, plus it saves the time of validating against a DTD as well as keeping the EDI message flexible. Since we detach the message from a DTD, we could change the format as and when required. We just have to extract the right data and push into the back end. In using DTDs, there's the overhead of changing the DTD, then changing the document to reflect the change. Instead, keeping the document well formed removes the overhead of maintaining DTDs and the need to send two files to the client side in case the EDI message is generated at the client.

There's a well-formed EDI message in XML format in Listing 2. One could use a DTD, parse the XML files and keep them

on the server as standalone XML files to be downloaded on request. The data could be populated in the elements and sent back to the enterprise where, since they're standalone, they could be used directly to extract the data and pump it into the back-end systems.

XML for Unstructured/Flat Messages

In the case of a flat format, like an HTML form, the submitted data would have to be structured explicitly into XML, then processed. Sometimes this is overkill if the amount of data is small, because there's the overhead of structuring the message into a particular format, parsing the document, then extracting the message.

XML is an easy way to demarcate data, but should be applied by considering the amount of data and whether it has any structure.

One can have XML documents conform to a specific DTD, which means that you can easily demarcate what documents are flowing into your system. The demarcation of data can be done using the document type. Hence the data can be extracted and placed in the appropriate queues to be handled by the appropriate application server.

Figure 1 illustrates the architecture in which the data to the enterprise could be received either through HTTP or SMTP.

USING HTTP

Through HTTP the data could come in to the Web server, which would load a servlet to service the request, then generate a Java Messaging object based on the type of message and route it to the appropriate queue.

USING SMTP

The messages could also come to the enterprise through SMTP protocol in which case the mail server would receive the message. A mail-retrieval service could be written via JavaMail and be used to look up the mailbox for mail received. The mail could contain headers that would indicate the presence of a particular type of message. The service could then extract the message that's sent as an attachment and, based on the header, push it to the appropriate queue to be handled.

The advantages of this approach are that when you can demarcate the messages, the following issues are resolved:

1. Since each message is handled by a particular queue, if one of the application servers goes down, the rest of

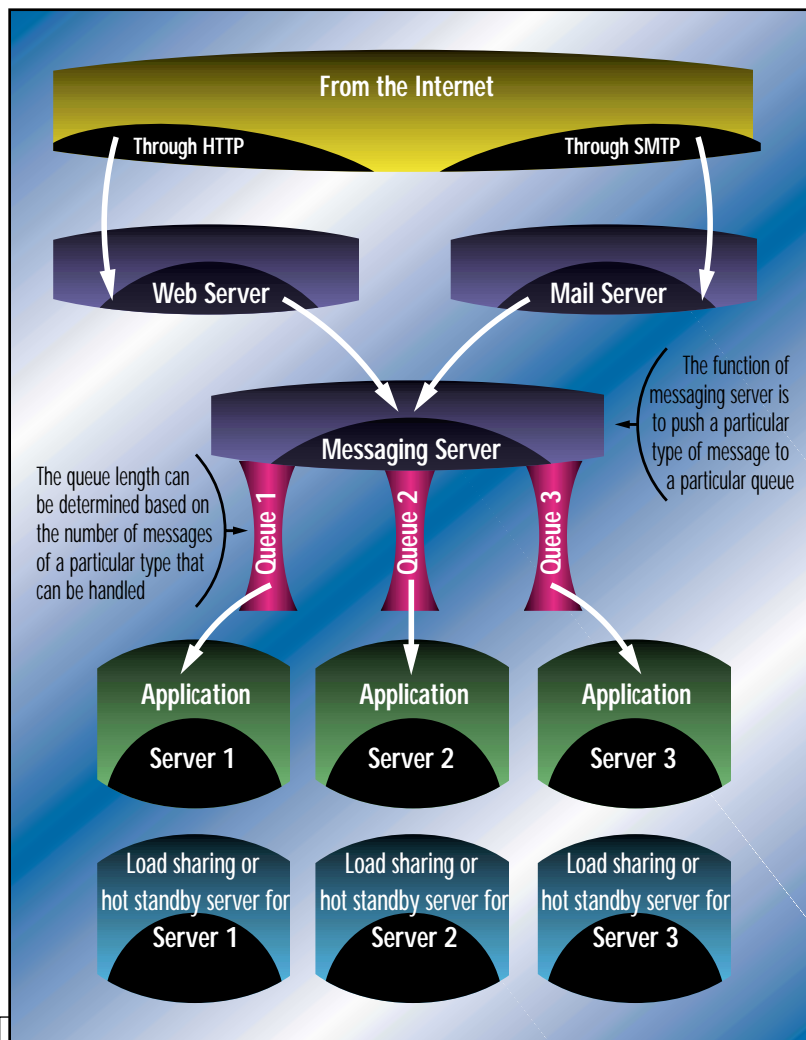


FIGURE 1 The messaging server divides the incoming messages (based on the document type) into different queues that are being listened to by different application servers. These servers can handle those types of messages.

AUTHOR BIO

Sandeep M. Nayak is a software analyst for International Object Technologies in New Jersey.

- the messages can still be processed.
- You can invoke other application servers for load sharing in case there's an increase in a particular type of incoming message.
- Log tracking and maintenance for each type of message becomes easier.
- It frees up the application server from having to poll constantly for incoming data to process.
- It implements a "push" model to account for which servers are sent data, thus saving critical CPU time for the servers.

Importance of Transactions in Messaging

Having transactions built into the messaging is an added advantage. Should the application server that's listening to the queue go down, the second-level application server, working as a hot standby, could connect to the same queue and start picking up messages that weren't handled or acknowl-

edged. Moreover, you could decide on the type of service, whether it's critical or noncritical, based on the type of message.

Most messaging systems have built-in transaction processing so they can cache unacknowledged messages and send them back to the server once it connects to the queue. The SpiritWAVE implementation of Java Messaging, for example, has such support.

The hot standby will constantly ping the application server that's servicing the messages. When the server breaks down, the hot standby will connect to the queue and start servicing the messages.

Either the servers could handle the processing of messages, or the (Message) objects sent by the logging server could encapsulate the logic to process the message data. In case the objects encapsulate the logic, all the message has to do is implement a known interface that the server will use

to extract the data and connect to back-end systems. These back-end systems could be any of the following:

- A database
- A legacy application, in which case it would have to format the data in a specific format
- A dump to a filing system
- Another application waiting for the data

Conclusion

XML has opened newer avenues to processing data in a simple and straightforward manner. XML documents' inherent property of maintaining structure has become the backbone of most messaging systems and eased the need to demarcate data and process each type differently. XML is looked on as a major technology in future messaging systems. 🌐



LISTING 1

```
<?xml version="1.0" standalone="YES"?>
<EDIMESSAGE NAME="STANDARD MESSAGE">

<UNH>
  <BGM>Beginning of message</BGM>
  <MSG>Message type identification</MSG>
  <RCS>Requirements and condition</RCS>
  <DII>Directory Identification</DII>
  <RFF COUNT="counter value">Reference to other information-
  al sources for this
message</RFF>
  <RFF COUNT="counter value">Reference to other information-
  al sources for this
message</RFF>
  <DTM COUNT="counter value">
    <CENTURY>19</CENTURY>
    <YEAR>99</YEAR>
    <MONTH>10</MONTH>
    <DAY>01</DAY>
    <HOUR>10</HOUR>
    <MIN>23</MIN>
    <SEC>21</SEC>
  </DTM>
</UNH>
<SEGMENTGRP ID="1">
  <PNA> Party Identification </PNA>
  <ADR> Address of the party </ADR>
</SEGMENTGRP>
<SEGMENTGRP ID="2" COUNT="counter value">
  <CTA> Contact Information </CTA>
  <COM COUNT="counter value"> Communication Contact :-
  Mode 1</COM>
  <COM COUNT="counter value"> Communication Contact :-
  Mode 2</COM>
</SEGMENTGRP>
<SEGMENTGRP ID="3" COUNT="counter value">
  <DFN>Definition of the function </DFN>
  <FTX COUNT="counter value" MAXCOUNT="100">Textual Infor-
  mation </FTX>
</SEGMENTGRP>
<SEGMENTGRP ID="5" COUNT="counter value">
  <SGU>Usage details Type 1</SGU>
  <FTX COUNT="counter value"> Textual information</FTX>
</SEGMENTGRP>
```

```
<SEGMENTGRP ID="5" COUNT="counter value">
  <SGU> Usage Details Type 2</SGU>
  <FTX COUNT="counter value"> Textual information</FTX>
</SEGMENTGRP>
<SEGMENTGRP ID="13">
  <AUT> Authentication Result</AUT>
  <DTM> Date of authentication </DTM>
</SEGMENTGRP>
<UNT messcount="8" controlrefno="a unique no"> This is the
end of the message and defines the total count of segments
in the message as well as a control reference number of
the message.</UNT>
</EDIMESSAGE>
```

LISTING 2

```
<!-- DTD for the attached EDI message -->
<!ELEMENT EDIMESSAGE (UNH,SEGMENTGRP*,UNT)>
<!ATTLIST EDIMESSAGE NAME #PCDATA #IMPLIED>
<!ELEMENT UNH (BGM,MSG,RCS?,DII,RFF*,DTM*,FTX*)>
<!ELEMENT SEGMENTGRP (PNA,ADR?) | (CTA,COM*) |
(DFN,FTX*) | (GRU,FTX*) |
(SGU,FTX*) | (FNT,REL?,(GIR,FTX)*)
(RFF,FTX*) | (ELU,(ELM,EDT)?,IMD*,GIS*,FTX*)
(MEA,FTX*) | (ELV,FTX*) | (CDV,FTX*)
(DRD,FTX*) | (AUT,DTM?)>
<!ATTLIST SEGMENTGRP (ID,COUNTER) #NMTOKEN #REQUIRED>
<!ELEMENT (UNT,BGM,MSG,RCS,DII,PNA,ADR,CTA,COM,DFN,
FTX,GRU,SGU,FNT,REL,GIR,RFF,ELU,ELM,EDT,
IMD,GIS,MEA,ELV,CDV,DRD,AUT) -- #PCDATA>
<!ATTLIST (COM,FTX,GIR,IMD,GIS) COUNTER #NMTOKEN #REQUIRED>
<!ELEMENT DTM (CENTURY,YEAR,MONTH,DAY)>
<!ELEMENT CENTURY #PCDATA>
<!ELEMENT YEAR NMTOKEN>
<!ELEMENT MONTH NMTOKEN>
<!ELEMENT DAY NMTOKEN>
```



Silverstream

www.silverstream.com

Dive into the XML Specification

REVIEWED BY TIJA RAGAS

XML: The Annotated Specification

by Bob DuCharme

368 pages, Prentice Hall

The name of the author of *XML: The Annotated Specification* may sound familiar. Some of you may have heard of him because of his contributions in the XML industry. For others it's probably because you've just seen his name on the cover of this magazine – Bob is writing the **Standard Watch** column. This book is an excellent sample of Bob's writing style and his ability to simplify the concepts introduced by a complex technology. I've never gone through the complete XML specification; however, like other language specifications, it's not for the average reader. I think Charles Goldfarb puts it succinctly in his foreword to the book when he describes the spec's terseness: "For earth people – even for most programmers – it's a daunting read indeed!"

Nevertheless, *The Annotated Specification* is clear and at the same time objective in its coverage of the XML spec. It is what it says it is – a specification book. It simplifies and clarifies the specification using pertinent examples, sidebars, tips and reviews of complex topics. It serves as a general reference for the XML language and a guide to the specification. The author goes through the specification line by line and then expands on each concept by offering insight into why the guideline/definition was created, what it means and how it's to be used.

This book isn't meant for readers who are trying to write XML applications or learning the XML language and its relationship to other technologies such as Java. It doesn't cover applications of XML in the industry, XML tools, and so on. That's not its purpose. The book is targeted to advanced readers who want to understand the reasoning behind the XML language and its pure definition. If you've never had a hankering to read the specification, then you probably won't get much out of this book.

The first section of the book, "Annotation Specification," consists of a couple of chapters that introduce the book and the XML specification. I believe these chapters will be useful to readers who aren't interested primarily in the specification. The information here may have been garnered from other resources, but may not be available elsewhere in such a comprehensive form. The author starts with the reasoning behind the birth of XML. He discusses XML in relation to HTML and SGML and

what roles these technologies play in computing. This introductory chapter has nothing to do with the specification itself, but will be of use to most readers. Many of the concepts and XML components such as XSL, XLink and Xpointer are covered along with XML's relationship to browsers and scripting. The author ends the chapter with a brief description of the purpose of the specification and the additional information that this book has to offer.

Chapter 2 provides a preface to the W3C XML specification, and gives pointers on where readers can pick up the specification, its versioning, its notation and syntax.

The next section of the book introduces the XML specification. The origin, goals and terminology of the specification are described in Chapter 1. One of the most useful features in this book becomes apparent here as the author takes each line of the specification and clarifies words that may be ambiguous. He clearly sets the context for the terms and gives the reasoning behind them.

An example is the author's explanation of what "parsed data" means in the context of the specification. Parsed data isn't data that has been parsed, but data for the XML processor (defined in the specification's next paragraph) to parse. Earlier drafts of the spec used the terms *text* and *binary* rather than *parsed* and *unparsed*.

These types of explanations are invaluable for eliminating confusion and ambiguity typically associated with language specifications.

Chapters 2–4 go over the main body of the specification and the XML language. XML documents, their components, DTDs, logical structures and their elements and attributes are covered in Chapters 2 and 3. Chapter 4 discusses physical structures, including entities, and introduces XML processors.

Chapters 5 and 6 focus on XML processing. Chapter 5 discusses validating and nonvalidating processors and their use. Chapter 6 elaborates the formal grammar rules for processing XML documents.

The remainder of the book comprises six appendices that provide further sources of information and explanations on related topics. A concise glossary for the terms used in the text is also provided.

A "specification" book isn't something that most folks read cover to cover and I'm no exception. However, it's a great reference to have when you need to know the meaning of various terms and concepts. ☪



Activated Intelligence

www.activated.com



Working hand in hand to allow processing of information from external systems

Java and XML – The Promised Land

Welcome to Java and XML – the promised land. In the context of these two technologies the promised land presents a series of solutions in which the marriage between Java and XML has provided an optimal answer for solving distributed multiplatform problems. But do XML and Java actually pave the road to data interchange nirvana? Let's embark on a journey to analyze and evaluate the hype and the reality of the solutions offered by the combination of these technologies. Throughout this journey we'll explore the architectural merits associated with deploying these solutions.

In this column I'd like to present how these two technologies, Java and XML, can be combined to help you achieve your own personal path to the promised land. Some of the areas we'll cover in this column are:

- The use of XML files as deployment descriptors for Java applications and their use as more sophisticated forms of property files
- The use and advantages of XML databases as persistence storage for Java applications and how Java programs can leverage them
- The use of XML tagging to describe the data content of JMS messages, i.e., a serialization and deserialization strategy for JMS messages
- The use of XML to describe the services provided by a mobile agent, i.e., the use of XML repositories as interface warehouses to discover the behavior contained by mobile agents

If you'd like to see other topics in this column, please e-mail me and I'll try to address them.

This article will introduce you to the strengths and characteristics of the Java and XML marriage. I'll concentrate on related technologies such as DOM, SAX and DTDs – technologies that help provide a comprehensive solution using XML and Java. In the following issues you'll see a lot more coverage on Java itself.

Java & XML

The Java platform has evolved to become the de facto standard when deploying enterprise-wide Internet-

enabled solutions. Recently, the introduction of the J2EE (Java 2 Platform, Enterprise Edition) has helped to standardize the application server environment by providing a reference platform for developers to build enterprise solutions. J2EE has extension APIs that address the development of distributed enterprise solutions. The set of APIs includes:

- JDBC for database access
- RMI for distributed Java communications
- Java IDL for distributed CORBA communications
- Servlets for Web server-side applications
- JNDI for name and directory service access
- JMS for asynchronous message communications
- EJB for developing enterprise application components in a distributed fashion

In the J2EE environment the support for IIOP over RMI (Remote Method Invocation), the evolution of the JMS (Java Messaging Services) and the evolution of adapters for the JavaMail API have created a stable communications pipe in which information can be exchanged reliably in a heterogeneous environment. This communication pipe provides a standard conduit for passing XML information.

In a very short time XML has evolved to become the de facto standard for data manipulation between enterprise- and Internet-enabled applications. The programmatic tools that have contributed

to the proliferation of XML technologies in enterprise applications are parsers. The two main parser technologies are the DOM (Document Object Model) and SAX (Simple API for XML) APIs. Other XML technologies have contributed to the acceptance of the language as well. Some of the most popular technologies are XSL for data viewing, DTD for data verification, XML databases for permanent storage, XML integration servers for business-to-business interaction and the evolution of HTML (4.0) to an XML-compliant format. We'll come back to DOM, SAX and DTDs later in this article.

The Java – XML Highway

Let's take the analogy of a highway. In its most basic form you can think of the Java platform as the provider of the highway and XML as the cars. The Java platform provides the bridges, intersections and tollgates for the cars to drive through. XML provides a standard definition of what a car is. One definition is that it must have four wheels, a chassis with at least two doors, an engine, a steering wheel and, at a minimum, one car seat with a seat belt. While these are concrete components, there are other ways in which these technologies can be combined to achieve ideal results. Imagine the Java platform as the automobile builder and XML as the blueprint for building the car. Using this combination, the output produced by the Java factory is controlled by the XML input but the output itself doesn't have to be XML specific. However, if we combine these two approaches, we can see how XML can be used as the blueprint in the Java factory, how the Java factory produces XML-defined cars and how those cars can move from one location to another using the Java-enabled communications highway. This highway can be implemented using a combination of e-mail, IIOP, JRMP or JMS (see Figure 1).

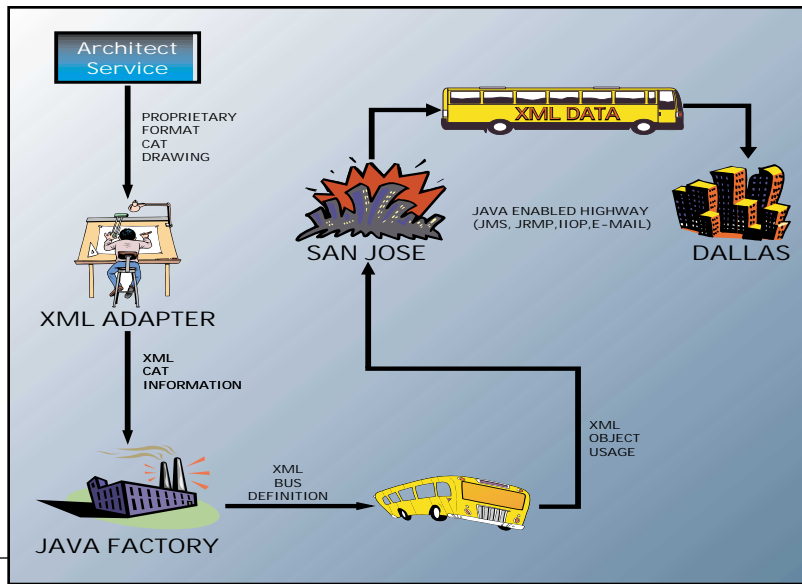


FIGURE 1 The Java communications highway for XML

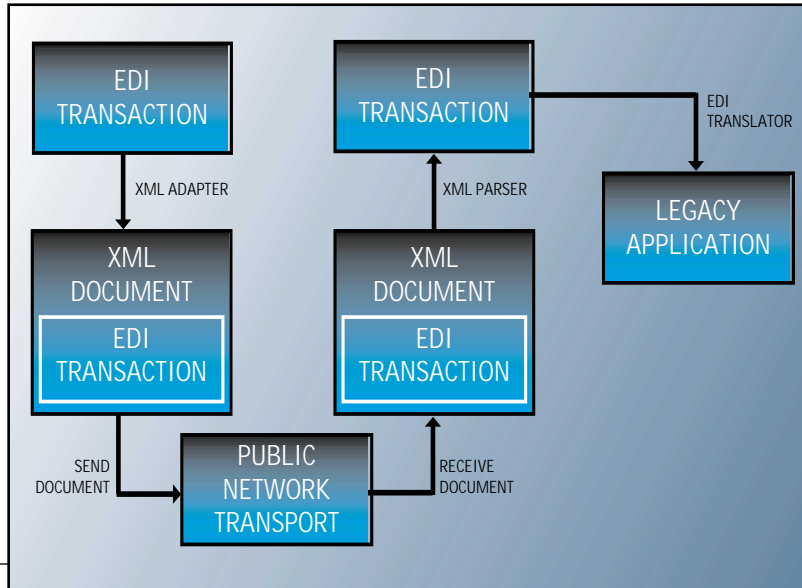


FIGURE 2 EDI transactions and XML documents

As you can see from these examples, XML can be used as input or output for another application while Java can be used as the producer or consumer of the data produced by XML.

Beyond Java

Up to this point I've concentrated solely on Java and XML – after all, that's the title of this column! However, the real power of XML is that its creation can proceed from any producer application; such applications need not be Java based. Similarly, the consumption of XML information can take place using applications that aren't Java based. Legacy applications can also share information with XML consumers and producers by leveraging the adapter

design pattern and providing applications that transform the legacy data into XML bidirectional streams.

Adapters allow applications or components built for one system to be used in another system. This is accomplished by creating a wrapper interface that acts as a translator of information between the new interfaces and the existing class interfaces.

One area where we can leverage XML-based adapters for manipulating legacy data is EDI transactions. EDI transactions are hierarchical formatted files that contain information encoded using standard predefined tags. XML can be considered a superset of EDI. By using XML as such, we can store additional information on how to handle the infor-

mation contained inside an EDI transaction (see Figure 2). This information can consist of sender as well as recipient information, and include details on the priority of the information, special care for instructions associated with the transaction's content and other items not normally stored as part of transaction information.

Back to Java

The Java language facilitates the creation of adapters by providing a platform that can be used on heterogeneous operating systems for parsing and creating XML documents. The two main APIs used for this purpose are the DOM and SAX APIs, introduced earlier. The DOM API provides a tree-like representation of the hierarchical data. The application code traverses the tree by accessing tags contained in the document hierarchy. Once the specific tag is encountered, the attributes, data and text associated with the tag can be accessed. Developers using the DOM are normally concerned with the hierarchy and structure of the document. The SAX API provides an event-driven mechanism that allows applications to search for a specific tag independent of the document hierarchy. One of the characteristics of this approach is that the only information available when a tag is reached is the name of the tag and its attributes. The textual information associated with the tag isn't accessible directly. Developers using the SAX are normally concerned with the tags contained inside the document.

There are clear situations when you need to use a hierarchical tree view of the document. This will be done using the DOM API. One of those situations is when you need to search information inside a document and the tags in the hierarchy contain semantic information. This would be the case if you were looking for all the models of Ford trucks that carry blue tones (see Figure 3A). Just finding the blue attribute inside the truck model won't be particularly meaningful. However, finding it inside the Excursion, Expedition and Explorer lets us know that these are Ford trucks with blue tones (see Figure 3B). Notice that the F-150 truck doesn't carry any blue attributes and thus isn't part of the resultset although the cars tag has a blue attribute.

In some situations you just need to know the occurrence of a tag. This is done using the SAX API. One situation in which the hierarchy of the document isn't of immediate importance is when

AUTHOR BIO

Israel Hillierio is a Sun-certified Java programmer with 10 years of programming experience, including three and a half in Java. He holds Ph.D. and MS degrees in computer science and a BS in computer engineering.

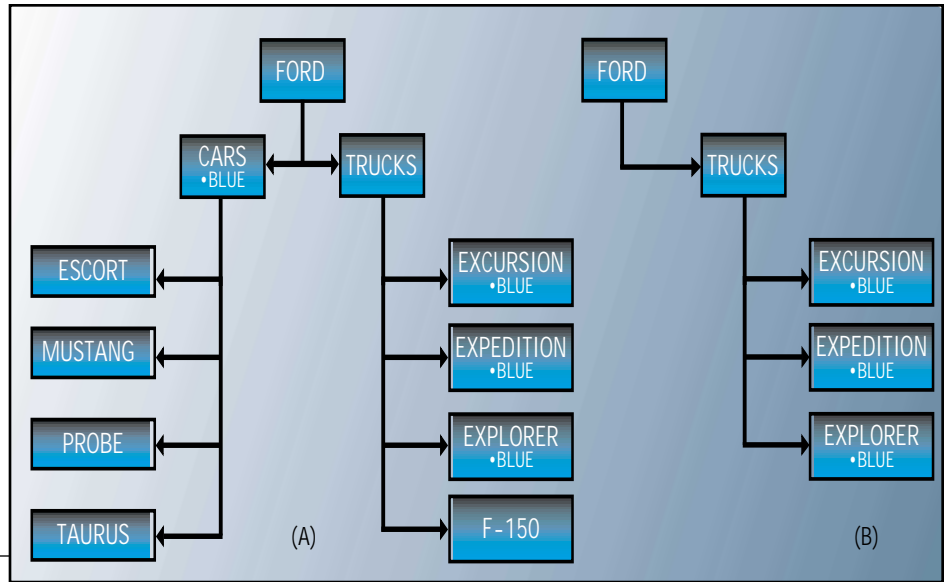


FIGURE 3 Ford models hierarchy is suitable for DOM.

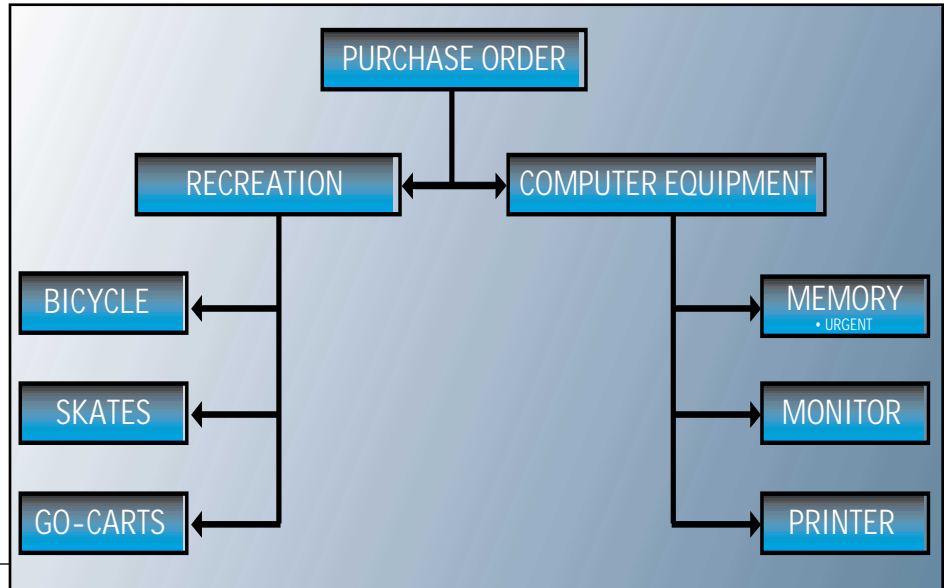


FIGURE 4 Purchase order hierarchy is suitable for SAX.

there are key tags that can trigger the execution of separate processes. This would be the case if you were looking for the Urgent attribute inside any item in a purchase order (see Figure 4). One of the items in the purchase order marked Urgent could be computer memory. It doesn't matter that the memory item is contained inside the computer equipment hierarchy. What is important is that the memory item has an attribute of type Urgent. Finding the Urgent tag in any of the items will allow the purchase order to be processed through a priority queue.

The two previous approaches can be combined to allow the distribution of work through different process paths based on specific tags' names or attributes (SAX API). This is similar to the processing of urgent orders through a specialized queue based on the attribute informa-

tion contained inside the item tag. Once the purchase order has been selected, the semantic information contained by the document hierarchy becomes valuable for processing the content of the order (DOM API).

In this column you've seen one of the major programmatic areas where Java and XML can work together hand in hand to allow processing of information from external systems. The XML and Java technologies are parser based and leverage document hierarchies and attributes to retrieve semantic information about the information contained inside the tag. Using these two approaches you can build systems that leverage workflow engines to tailor the processing of information. ☛

Elixir Technology

www.elixirtech.com

Exce

www.odi.co

eleon

om/excelon



What you need to know about XML as a metalanguage

XML in the Enterprise

Welcome to <e-BizML>! Some of you may be familiar with my *e-Java* column in SYS-CON Publications' *Java Developer's Journal*. I'd like to continue <e-BizML> in the same flavor by focusing on the business aspects of XML in the computing industry. Similar to *e-Java*, I'll offer my two cents on an XML (or related technologies) book in this and subsequent issues. I want to make this column as interactive as possible and would appreciate any feedback from you, the reader.

Let's talk about what we're going to talk about in this column in the coming year. We'll discuss, for example, XML's role in e-business today, and new and exciting events taking place in the world of XML. And as they emerge, I'll provide overviews of specific new technologies and discuss their impact on the XML industry. Our focus here is on XML in e-business – hence the moniker <e-BizML>.

I'd like to start by examining how the term *metadata* is used in the industry and how it relates to XML. We'll look at some of the common areas in distributed enterprise applications that can leverage XML and related technologies. We'll also look at an organization that helps fund XML projects in the computing industry. I'll end with a review of the e-book for this month – *XML Pocket Reference*.

Let's Start with Metadata

Metadata is often referred to as “self-describing” data. In other words, it's data that describes data. It defines a common language that allows data to be shared among people, systems, processes and programs, resulting in more effective communication. All organizations have their own terms and definitions that facilitate communication. However, the same term may mean something else in another context. For example, a “chip” refers to something edible (very much so) in the food industry, and something different in the electronics industry. At the same time, different terms in different contexts may be used to refer to the same entity. For example, a “Customer” in one organization may be a “Client” in another.

When different terms are used to refer to the same data, interpretation and maintenance of that data becomes complex. Metadata defines a common language

used within an enterprise or an industry consortium so that processes and programs within that enterprise (or industry consortium) can communicate using a common base for the data definitions. For example, the title, release date and singer's name constitute metadata that describes a music album or CD. The songs themselves may be viewed as the content or the data.

A Metalanguage for Defining Metadata

Markup languages provide a means to document metadata in computing. XML is a markup language used to create conceptual documents in the form of character strings. The format of the document is defined by marking up the content of the document based on a set of well-formed rules. Metadata is used to define the structure of an XML document or file. The language rules are in the form of a DTD (document type definition) that in turn follows the rules of XML. As XML is used to define other languages (such as DTDs), it is thus, by definition, a metalanguage (a language that defines other languages).

XML provides a much-needed standard in the way firms exchange and present information over the Internet. It does so by facilitating computer-to-computer communication via the use of standard data formats. These formats are standard across a specific business realm or environment. Metadata that's used by various industries can be used within XML to define markup vocabularies. XML serves as an enabling technology that facilitates integration of structured and unstructured data for e-commerce applications.

Applying XML

Okay, XML is a neat technology that allows us to communicate using standard

data formats. That doesn't say much for how it can contribute to business applications. In a nutshell, XML promotes an extensible environment in which data can be exchanged in a standard manner. This can contribute to a variety of enterprise application areas. Following are some of the areas in business computing in which XML is currently making an impact. They're listed randomly, without regard to any order of importance.

- **Deployment descriptors for runtime:** A natural fit for XML is its use as a mechanism for defining deployment descriptors for an application's runtime environment. XML can be used to replace the paradigm of configuration files that are hard to maintain, static and not very portable. Software components can be deployed with accompanying XML descriptors that offer more sophisticated runtime capabilities by using XML's metadata description capabilities. For example, this mechanism has been adopted by the Enterprise JavaBeans deployment model.

- **Information distribution:** The Web has created a revolutionary way of disbursing information from disparate sources to a multitude of participants in a distributed environment. This is because it offers a pervasive distribution channel by leveraging the Internet. XML technology is based on the premise that content and presentation should be separated when exchanging information. This makes it easy to distribute the content and leave the presentation of the content up to the client application. Of course, XML provides the mechanism (XSL) for data presentation also. Nevertheless, end applications have the option of processing the content without presenting the data in a browser. DTDs that accompany the data enable applications to offer more structured and “intelligent” information.

- **Enterprise data management.** Enterprise applications require the exchange of heterogeneous data; it may come from different data sources, in the form of various formats and transferred using a variety of communication protocols. This leads to a substantial requirement to integrate the data so that applications can use it. XML provides a universal representation of data for information exchange between applications. It does so by facilitating data sharing and communications between different applications and customizing the presentation of the data.
- **Business transactions and data transformation.** Besides being the Web standard for exchanging data, XML is also becoming the standard for business transactions. This is especially true in legacy environments. For example, whenever data needs to be migrated between dissimilar systems, data transformations are required. XML offers a standard format for transporting data in the middle tier.
- **Business process workflows and data integration.** Business process workflows consist of sub-workflows and processes from various organizations within the enterprise. This leads to a requirement to integrate the data from these processes. The data may be in different formats. XML offers a viable option as an integration technology in that it facilitates data interchange. Standardization of XML-based workflow systems will ensure that different workflow systems can exchange business process information using XML as a medium for data exchange.
- **Knowledge management.** XML acts as a facilitator in knowledge management by coordinating the interpretation of disparate data. XML DTDs provide a mechanism for modeling knowledge as well-formatted structures. The extensibility of XML and the flexibility of DTDs enable application developers to transfer knowledge between different applications using standard data interchange formats. This also enables applications to export data from existing legacy document formats and to extend these documents by wrapping them with metadata, which is defined using XML.
- **Searching and pattern matching.** XML enables information to be accessible in a highly structured form. Its extensibility lets applications define custom tags that enable more sophisticated and complex searches. Again, the mechanism for this is the metadata that can be expressed

using XML. XMLs enabling the data to be described in a hierarchical structure also enhances the capabilities to recursive searches, parametric searches, and so forth.

- **Application integration.** XML provides the infrastructure for inputting and outputting documents containing metadata. This metadata promotes a common vocabulary for data interchange between different applications, thus enabling them to integrate with other applications. XML also facilitates information aggregation by defining common data formats in which information sources can receive and aggregate data from various sources.
- **Personalization and content management.** XML enables presentation of customized content for different users, depending on their preferences and business interactions. XML enhances the personalization capabilities of e-commerce applications because it's very effective in managing structured information. Indeed, that's the salient functionality offered by this technology. Thus XML can be used for creating and managing data structures that can be customized effectively for different users. The structures can be reused in different environments, which makes any proposed solution inherently scalable. In a similar manner, XML supports data syndication, content replication and management.
- **Messaging and data transport.** A misconception regarding XML is that it's also a means for data transportation. A more accurate description of the role played by XML in data transportation is that it facilitates the definition of common message formats that allow messages to be exchanged between different applications. Consequently, XML can make use of exchanging transport protocols for data interchange.

Funding XML Projects

Although XML as a new technology has garnered all the hype typically associated with computer-related technologies, the computing industry is still struggling with areas in which XML can be leveraged to provide solutions. Since XML impacts several areas of computing, several innovative ideas for leveraging its capabilities will start emerging in different organizations within a company. When a technology is at such a stage, it always helps if it finds a patron who will nurture ideas and help them grow. And it always helps if the patron backs this up financially.

One organization that helps fund XML projects is XMLFund, a venture fund in Seattle formed in 1999 by David Pool, a veteran in Internet start-ups. I hope to interview David in a forthcoming issue of *XML-J* so readers can learn more about the fund and its mission. This organization plans to invest exclusively in companies that are working with XML technology. It's based on the same principles as the Java Fund, an organization funded by California-based Kliener Perkins Cauffield & Byers that focused exclusively on Java technologies. Some of the companies that have been funded by XMLFund are Nimble.com, Digital Counterpart, and PhotoTrust.com.

Marking Up

XML, in my opinion, is going to be a key technology that will help the computing industry meet the demands of e-business for the next generation of computing. Its acceptance or rejection will depend heavily on how XML standards mature in the next year and how widely they're accepted. I hope this column helps you understand the role, developments and acceptance of XML in e-business in the years to come.

<e-book>

This month's XML book, *XML Pocket Reference* by Robert Eckstein, is the first one I've looked at in O'Reilly & Associates' Pocket Reference series. At first glance it's what it says it is – a small, pocket-sized reference that's easy to carry in your pocket or organizer. It's a well-organized and concise reference for basic XML concepts. After a brief introduction to XML and its relationship to HTML, the author defines XML terminology and the basic structure of XML documents. Chapter 2 is a reference to XML itself; Chapter 3 serves as a reference for DTDs. The remaining chapters cover XSL, XPointer and XLink. The coverage is concise, to the point, and written in a clear, easy style.

Pocket Reference is an introductory text for the reader new to XML. It may also serve as a reference when you're going out to a technology discussion and want some conceptual clarifications. To my mind, that's what this book should be used for. A reader familiar with XML will probably need a reference with more detail, examples and detailed descriptions. However, at \$8.95 and 107 pages, you won't lose anything by keeping this one handy as a quick reference to XML concepts. 🌐





Interview...

with

COCO JAENICKE

MARKETING MANAGER AND XML EVANGELIST
EXCELON CORPORATION



XML-J: Would you care to comment on the state of XML technology in the industry today?

Jaenicke: The official "state" of XML is that it's been accepted, but I don't think it's well understood. Most IT managers and project leaders have XML on some checklist somewhere, but few have yet incorporated it in a strategic way.

What's most interesting about the state of XML – past, present and future – is the direction that it's moving. Technology (consider Java) usually comes from the Ivory Tower, and it eventually pushes its way into the mainstream. XML is completely different – it has actually been pulled into the mainstream. And the speed at which it's being accepted is also head-turning.

XML-J: I heard you have some very exciting news to share with our readers regarding eXcelon.

Jaenicke: eXcelon Corporation, formerly Object Design, Inc., changed its name in order to shift emphasis to the fastest-growing part of its business.

XML-J: Does this reflect the market view that OO databases aren't going to survive in the B2B market? Last year I was under the impression that eXcelon was just a business unit spawned off by ODI. Now it seems like you're making XML servers your main business.

Jaenicke: OO databases will certainly survive – Object Store is fabulous technology – but only in a limited subsection of the B2B market. We're still continuing to support and grow that business, but you're right about the change of business strategy – eXcelon used to be a small, start-up interest on the side. Now it's the center of eXcelon Corporation's main business strategy.

XML-J: What advantage do you think eXcelon has over the competition in terms of market strategy and positioning?

Jaenicke: eXcelon Corporation has a unique differentiator that's based on our past successes. We believe any solution – B2B or other – should be flexible and dynamic in order to easily support changing business objectives. Our B2B solution is designed to be dynamic from the ground up. It's integrated with your business process so it's possible to fully leverage any and all partners in a way that complements your immediate needs.

XML-J: What does your product line consist of?

Jaenicke: Our B2B product line consists of three offerings: a Dynamic Application Platform for building portals and e-markets, a B2B Integration Server for enterprise B2B infrastructure and eSolutions for industry-specific frameworks.

XML-J: How can our readers start using your products? What type of individuals do you think can immediately leverage your technology?

Jaenicke: eXcelon B2B and B2C solutions can be used today – many diverse groups have – check out our Web site at www.exceloncorp.com.

XML-J: WebMethods is currently the name associated with the term B2B Integration Server. Are you stepping on their turf? How does your approach differ from WebMethods?

Jaenicke: We certainly compete with their product. The difference is our solution is dynamic, meaning you have complete control over whom you work with and how you work with them. eXcelon B2B doesn't require specific software to be installed on the partner's end and doesn't dictate vocabularies or protocols, so organizations can work cooperatively, not coercively.

XML-J: There's a lot of buzz about XML servers and portals. What exactly is an XML dynamic application platform and how does it relate to XML portals?

Jaenicke: A dynamic application platform hosts business logic that's built for change. Because of the extensibility of XML, it's possible to build and deploy applications using dynamic data modeling. This is key for portals because they're information driven and that information – even the format of that information – is always changing.

XML-J: Do you consider yourselves market enablers or marketplace creators? For example, will people use your products to create marketplaces/exchanges on the Web? Or are you providing the marketplace and signing on parties into a trading community?

Jaenicke: Both, because we have a range of product offerings. The eXcelon Dynamic Application Platform is for building any type of portal, including an e-marketplace. With the eXcelon B2B Integration Server, we're enabling organizations to partner with anyone, including short-term relationships with any e-market. And that's where having a dynamic solution becomes imperative – you can't coerce an established marketplace to install software or use your protocol. You have to be able to walk up to a desirable partner and say, "How do you do business? Okay, we'll work your way."

XML-J: Are you participating in any standards bodies, consortiums, etc.?

Jaenicke: Yes – we're members of the World Wide Web (W3C) Consortium, RosettaNet and OASIS.

XML-J: Do you plan to define any XML standards?

Jaenicke: Definitely not. We believe in being dialect agnostic because there will always be new industry vocabularies and protocols, as well as customizations. Look at how often EDI was customized. For that reason we believe the best approach is not to depend on there being a single standard, but to support the extension of standards.

XML-J: What vertical and horizontal market segments do you plan to target?

Jaenicke: Initially, eXcelon Corporation will be targeting the insurance, retail, telecommunication and manufacturing verticals, but I certainly expect to see that list grow over time. Horizontally speaking, any company that's building a portal, auction site, e-market or any B2B infrastructure will be interested in our technology.

XML-J: What areas do you think eXcelon will expand into beyond its current offerings? What kind of vendors do you see yourselves partnering with?

Jaenicke: We're partnering with enterprise application integration (EAI) vendors as well as those that have XML tools or expertise.

XML-J: How big a market are you going after? How much of it do you hope to capture?

Jaenicke: Using various different predictions and stats, we expect there to be \$6 billion in spending on B2B software infrastructure by the year 2003. I'd sound defeatist if I didn't say we were aggressively going after a large portion of that!

XML-J: Where do you see the XML market going in the next five years?

Jaenicke: At the risk of sounding antagonistic, I'd argue that there really isn't an XML market per se. XML is a technology and we're on the brink of an explosion of tools and applications that leverage XML's unique benefits. Five years down the road I expect XML will be fully built into virtually every solution – you may not even know it's there. ☺



SD 2000

www.sdexpo.com



A look at the history of DTDs

Replace DTDs? Why?

Of all the standards to accompany XML that are currently in progress at the W3C, few are more anxiously awaited than the Schema standard – the specification that provides an alternative to XML 1.0 DTDs as a way to describe a document's structure. But what's wrong with XML 1.0 DTDs? How many alternatives have been proposed, and by whom? Why didn't the W3C address these concerns in the original XML 1.0 specification instead of waiting until now? I'll answer those questions in this column, and in my next column we'll look at the current state of the W3C Schema Working Group's unfinished proposal.

What Can They Do?

Just as a compiler can process the source code of a particular programming language more effectively if the program's data structures are declared up front, an XML processor is more efficient if it knows what kind of data structures to expect before it begins reading a document. XML 1.0 DTDs – which I will hereafter refer to as DTDs, although technically schemas express DTDs as well – can have five or six kinds of declarations, depending on whether you consider comments to be declarations (the XML spec is vague on this point):

- **Element type declarations.** An element type is a named class of elements, such as `h1`, `img` or `p` in HTML or `para` or `listitem` in the DocBook DTD.
- **Attribute list declarations.** An attribute list declaration lists the attributes for a given element type. The attribute list for HTML's `img` element type includes the `src`, `alt` and `align` attributes.
- **Entity declarations.** Entities name collections of information that a DTD or document can reuse elsewhere. An entity may represent a single character of text, a string of text or a complete file sitting outside the DTD.
- **Notation declarations.** When a DTD declares an external non-XML, or “unparsed” entity, it must identify the entity's format. A notation declaration tells the processor: “Here's a legal format for this document type's unparsed entities.”
- **Comments.** These look just like they look in HTML: `<!-- like this -->`. This is information for the parser to ignore.

What's wrong with these?

Weak Data Typing

The most common complaint about XML from people who come to it from the database and programming worlds (as opposed to those coming from the SGML publishing and HTML Web design worlds) is the lack of data typing. When these developers declare or define a named piece of information – for example, a field in a database or a variable in some Java or C++ code – they're accustomed to naming its type and then assuming that the processing engine underneath their application will ensure that any information stuffed into that slot conforms to that type. Once they declare an XML `Quantity` or `RetailPrice` element type, they don't want to write extra application code to ensure that the strings between the start and end tags really are integers and currency figures. Extra error-checking code isn't just annoying to write; it adds fat to the thin clients that XML is supposed to be so great for.

This wasn't a big deal in the SGML world because nearly every application was a publishing application. With XML's popularity in e-commerce development, data values like quantities and especially prices become more important. Although XML 1.0 offers a few types that help constrain attribute values, classic types such as integers, real numbers, Booleans and dates aren't among the choices, and application developers need them for element content as well as attribute values.

Document Structure Not Stored in an XML Document

DTD declarations have their own syntax that, despite using the “< >” angle brackets, is quite different from XML document syntax. Many newcomers to XML ask why XML isn't used to represent the structure of its own documents. The original answer was that XML was designed to be completely compatible with SGML, which had a larger base of applications and tools than most new XML users realize. These applications and tools played a big role in XML's initial jumpstart.

Since then, a revision to the SGML standard allows for legal SGML documents without the DTD declarations used to specify document structure – that is, to have what the XML world calls “well-formed documents.” If an XML document with no DTD can still be a legal SGML document, then the primary reason for using SGML DTD syntax no longer applies.

Another argument against specifying DTD structure with XML elements was that it would be confusing to include elements that describe other elements right in there with the elements that they describe. As it turned out, no one does this anyway; schema documents are always kept separate from the documents they describe, and documents point to their schemas with a processing instruction, a namespace declaration or some other mechanism.

Using XML elements to describe document structures has several benefits. It makes these structures much easier to develop because you can use any XML editor to edit and manipulate them – and I mean any XML editor, even the lame ones that merely dump your document to a visual tree and then write that tree back out when you save your document. (Paragraphs of text like the ones you're reading here are very cumbersome to edit on such an editor, but a schema document is naturally treelike.)

Application development is also easier for documents whose structure is stored in a well-formed XML document, because applications have easier access to information about document structure. SAX and DOM, the two current XML API standards, offer very little to an application that wants to check DTD information such as an attribute's declared type or whether a particular element is optional. With document structure definitions stored in a DOM tree or triggering the same SAX events that the document's elements trigger, an application can find out all it wants about that structure.

No Inheritance

A key reason for XML's popularity among system developers is its ability to easily describe fairly complex data structures. You don't have to squeeze everything into tables; if you like, you can represent a data structure as a hierarchical tree or, with the help of ID and IDREF attributes, as a directed graph.

One of the great features of the object-oriented world is the ability to define data structures as extensions of existing structures. With a well-designed hierarchy of object classes inheriting from each other, simple changes can affect as much or as little of this hierarchy as you wish.

Developers with object-oriented experience appreciate XML's ability to define and manipulate complex data structures, but they know that specifying every detail of every data structure from the ground up isn't the most efficient way to develop a system. They want a way to base a new element type on an existing one.

Potential Messiness of Parameter Entities

A parameter entity is a string of text or an external file that's been named so that it can be easily plugged into DTDs. The former, an "internal parameter entity," may contain a few attribute declarations that you can reuse in the attribute list declarations of several element types; an external parameter entity could be a file whose declarations will be used in multiple DTDs.

To keep the design of complex DTDs modular and maintainable, internal parameter entities sometimes build on each other in multiple layers, leaving you with references to parameter entities that have parameter entity references themselves – and those may refer in turn to parameter entities that contain more parameter entity references. Because it's all implemented using string substitution, it can get messy quickly.

Specialized data structures suited to each of these purposes would give developers more robust components to mix and match when building a document type's structure.

Weak Self-Documentation Facilities

As with XML documents – and, for that matter, HTML documents – you can put comments in DTDs that the processor will ignore by putting them between the `<!--` and `-->` delimiters. Like anyone else defining data structures, DTD authors have been encouraged to use these comments to explain the use of these data structures, but in keeping with tradition, they often skimp on this duty. Utilities do exist that compile reports on DTDs by examining the sibling and parent relationships of the various element types, but serious automation of documentation generation can only go so far because of the lack of clues about each comment's purpose. Java, on the other hand, offers the `@fieldname` notation to identify specific fields of information in the header of a class or method's source code, making it easier for an automated utility such as javadoc to create useful documentation easily with no human intervention.

It's ironic that Java is better than XML at allowing automated documentation generation, for two reasons. First, a big factor in the popularity of SGML was the way it easily let developers create systems that automated the creation of print, Web, WinHelp and CD documentation. Second, the original idea for XML, like Java, came from Sun; it was Sun's Online Information Technology Architect Jon Bosak who put together the W3C Working Group that devised a simpler version of SGML that would work more easily over the Web.

Replacement Candidates

Three groups of W3C member companies and a mailing list devoted to cutting-edge XML issues each assembled alternatives to XML 1.0 DTDs and submitted them to the W3C. Each proposal addresses some or all of the problems described here. Just as schemas express DTDs as much as the SGML-like XML 1.0 style does, XML 1.0 DTDs also qualify as "schemas," but in common practice people refer to the XML 1.0 way as "DTDs" and the new ways as "schemas." In addition to the W3C's Schema proposal, you may have heard of eight other schema proposals, but really only four were submitted – other names refer to earlier names or subsets of these four.

A group of eight authors, five of whom worked for Microsoft or DataChannel (a

Redmond company that's done a lot of XML work with Microsoft) submitted the XML-Data proposal to the W3C on January 5, 1998, making it the only proposal to predate XML's ascent to Recommendation status. A simplified version of XML-Data known as XML-Data Reduced, or XDR, was submitted to the W3C on July 3, 1998. On Microsoft's Web site XDR is also known simply as "schemas," with no mention of its full name, greatly adding to the confusion over schemas. Just remember that when Microsoft literature describes the use of schemas with IE5 or BizTalk, they mean XDR.

Microsoft, IBM and independent consultant Tim Bray submitted the Document Content Description (DCD) schema proposal on July 31, 1999. It expresses document structure using the XML-based Resource Description Format (RDF). While neither Microsoft or IBM has shown any interest in following up with DCD or even RDF since then, Object Design's (now eXcelon Corporation) eXcelon product still uses the DCD format to store its own schemas.

Before e-commerce software developers CommerceOne acquired Veo systems, developers at Veo submitted "Schema for Object-Oriented XML" (SOX) to the W3C on September 9, 1998. True to its full name, SOX makes mapping between element type declarations and object-oriented data structure definitions simpler and more straightforward than its predecessors do. The SOX proposal's frequent use of the term *electronic commerce* gives another clue about what kind of application development concerns drove its design.

Finally, the xml-dev mailing list that gave the world the Simple API for XML (SAX, the standard event-driven API to XML documents) also submitted the Document Definition Markup Language, or DDML (also known as "XSchema" and "XSD" along the way), on January 19, 1999. Although no one ever implemented it, DDML indicated to the W3C where an important group of XML developers saw the priorities in schema language development.

After receiving these proposals, the W3C took authors and editors from each of them and assembled a working group to put together their own schema proposal. After publishing a requirements document in February 1999, they released the first draft of their two-part proposal in May and the most recent in December. In my next article we'll take a look at some of the features in the W3C's proposal. 🌐

AUTHOR BIO

Bob DuCharme is an assistant vice president at Moody's Investors Service, where he oversees the implementation of SGML and XML systems. The author of XML: The Annotated Specification published by Prentice Hall, Bob received his master's degree in computer science from New York University,

 DUCHARMR@MOODYS.COM

XML FEATURE

BUILDING DISTRIBUTED APPLICATIONS WITH CORBA AND XML

Can these
technologies
benefit from
each other?

XML and CORBA are key technologies for building distributed systems, but both have evolved separately by addressing different needs of content management and distributed applications. This article illustrates how these technologies can benefit from each other.

XML

XML and related specifications are addressed on the W3C Web site (www.w3c.org) and in other articles in this journal. This section looks at the structure of an XML application as depicted in Figure 1. The entity manager reads XML documents from some virtual storage – a database, a file or some other mechanism – and creates XML *entities*, a term used to describe XML elements. This data is passed to an XML parser, which can optionally validate the document using either the DTD or, in the future, one of the semantic specifications (e.g., DCD, RDF). The application code accesses this data using the DOM or similar API mapping to some programming language to process the data. Since XML today is still about syntax, this code contains the logic needed by the semantics to make meaningful use of the data. As seen in Figure 1, XML doesn't have transport built into it. Availability of a transport facility is thus a requirement for building a distributed XML application.

CORBA

The common object request broker architecture (CORBA) is a specification by the Object Management Group (OMG). Figure 2 depicts the structure of a distributed application built using CORBA. Objects are described using the Interface Definition Language (IDL) and are distributed on a communication bus, the Object Request Broker (ORB). These objects are then implemented in a programming language, typically Java or C++, and the ORB transparently handles all network, platform and language issues. Additionally, the objects can be made transactional, secure, and so forth, using CORBA services. Other vendor-provided tools help in the management and administration of these objects, enabling the development of a large-scale distributed system. (Thousands of such CORBA applications have been deployed all over the world.)

A distributed application consists of three logical components – data, logic and transport. CORBA ties these three components together in the form of distributed objects in which the logic is implemented using some OO language. The data is tied to the transport and transferred in binary form between the client and the server. The familiar function call paradigm is used to invoke methods on remote objects transparently. In contrast, XML is only about data, which is represented as text and not inherently tied to any transport protocol. It can be carried over any transport protocol that's capable of transporting textual data. The XML programming model consists of walking a parsed XML document and taking action based on the elements encountered. Using XML for data gives us the ability to describe arbitrarily complex data types, build loosely coupled systems, easily transform data from one form to the other and easily display data in browsers. XML parsers are available for most languages and operating systems; thus it's easy to embed a parser with an application. XML support is also present in the popular browsers, giving developers the ability to display XML data. CORBA has been used successfully to interface with legacy and ERP systems. Using XML instead of IDL types may, in certain cases, make this integration even easier. Many ERP vendors have pledged XML support in their products, which would make the use of XML even more attractive.

A distributed system that uses CORBA for distribution and XML for data gives us the best of both worlds, and at the same time builds on two open industry standards. The following section looks at the specifics of how this integration can be achieved.

Integrating XML and CORBA

There's some commonality in the concepts of XML and CORBA, as illustrated in Table 1.

XML documents, which are textual in nature, define two popular programming APIs:

- **The Document Object Model (DOM) API defined by the W3C:** This API provides access to the XML document independent of the vocabulary of the document, that is, the API methods are generic document methods. It creates a full document object from an XML document, and thus may require substantial space if you're dealing with large documents.
- **The Simple API for XML (SAX) API:** This is an event-driven API and only the sections of the document that are of interest are loaded and presented for use.

CONCEPT	XML	CORBA
Data	XML Document	CORBA Primitive types, Structure, Sequence, Any, DynAny, value types
Type Specification	XML DTD, XML Schema	CORBA TypeCode
Namespace	XML Namespace	CORBA Namespace

TABLE 1 XML and CORBA: Common concepts

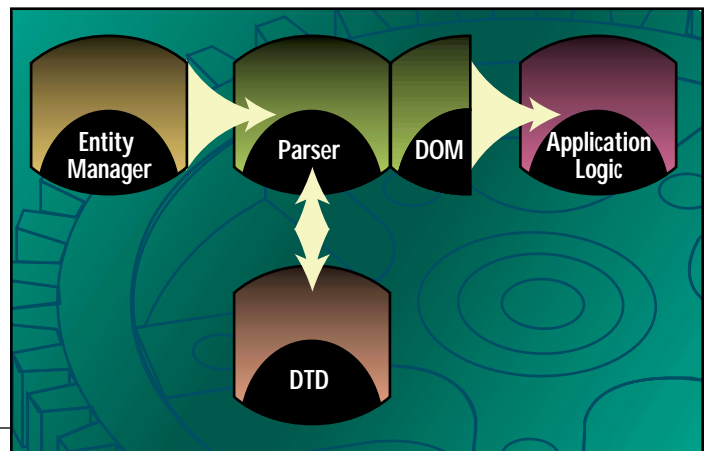


FIGURE 1 XML application architecture

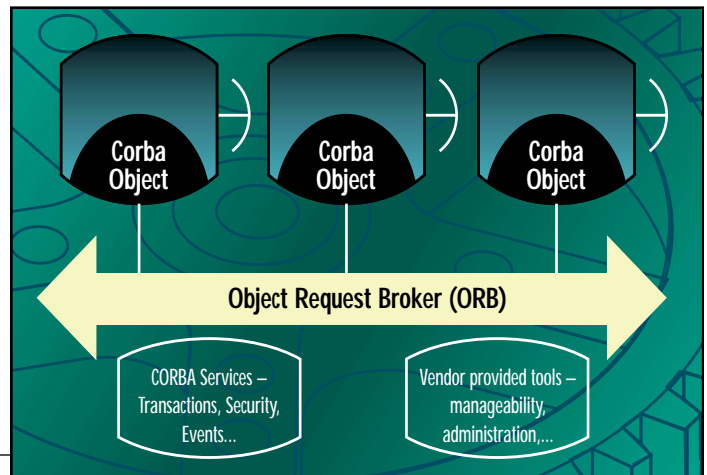


FIGURE 2 CORBA architecture

As both APIs are generic in nature, an alternative approach would be to provide an API in which the vocabulary of the XML document is mapped directly in the API. For example, the document is accessed directly using the element and attribute names that appear in the document. This way, the semantics of the XML document are visible to the code that manipulates it, resulting in greater legibility.

The interfaces of CORBA objects and the types of data they can manipulate are defined using IDL. Thus the integration of XML with CORBA must be expressed using IDL types. However, it's possible to perform this transformation dynamically when IIOP-compatible structures can be created directly from XML documents.

IDL provides standard primitive data types along with structured and aggregate data types and variable types in the form of the CORBA Any (and DynAny). IDL distinguishes between object types and value types as noted in the sidebar below.

It's not desirable to produce an XML IDL API using CORBA object types. This would imply that the XML document is encapsulated within a CORBA interface, which would be inefficient since the document is accessed in a remote fashion.

It's more desirable to use either value types or existing IDL data types as a means of passing documents by value, since this would allow efficient operations on the XML document. However, the value-types specification has only recently been ratified by the OMG, and few implementations of the specification are available. Thus the choice remains about whether the integration of XML and CORBA should use value types or map directly to existing CORBA data types.

Approach 1: SIMPLE APPROACH

This initial approach is the simplest one possible for passing an XML document to a CORBA object and involves "stringifying" the XML document.

```
struct XMLDocument
{
    String dtd;
    String document;
};

typedef sequence<XMLDocument> XMLDocuments;

interface MyObject
{
    void invoke(XMLDocuments documents);
};
```

One advantage of this approach is that it's easy to extend existing object interfaces to support XML data. The disadvantages are significant and are as follows:

- **Inefficient use of space.** In a distributed application the verbose XML document is distributed over the network.
- **Inefficient use of time.** The XML document must be parsed at each point of use.
- **Not type-safe.** There's no way of validating the XML document at transmission, and it must be validated at each point of use.

Approach 2: MAPPING XML TO IDL USING VALUE TYPES

This approach uses CORBA value types and is under review by the

OBJECT TYPES AND VALUE TYPES

When a CORBA object A invokes a method on a remote object B and passes an *object type* C to object B, a *reference* to the object type C is passed to B. Object B can subsequently invoke methods on C. These methods are remote invocations, since only a reference to object C was passed to B.

When a CORBA object A invokes a method on a remote object B and passes a *value type* C to object B, the *value* of the object is passed to B. Object B can subsequently invoke methods on the object. These are local invocations and thus very efficient.

“It's more desirable to use either value types or existing IDL data types as a means of passing documents by value, since this would allow efficient operations on the XML document”

OMG. It represents as much of the DOM API as possible using CORBA value types. The value type, a new IDL construct, is a cross between the existing struct and interface. It's always passed by reference, but can contain method declarations such as an interface.

Listing 1, taken from OMG Document 99-12-05, demonstrates the value-type definition for a DOM Node. The Node type is central to the DOM API, since the document, element and text sections of an XML document implement the Node interface. As you can see, this API uses a standard naming convention for accessing all parts of an XML document.

Other Key DOM types are DOMString, NodeList, Document, Element, Attr and Text. This approach is consistent with the standard DOM API; however, the semantics of the XML document aren't clear when using a standard API. It's also likely to be some time before such an approach is practical, as the CORBA implementation of the value-type specification hasn't become common yet.

Approach 3: MAPPING XML TO CORBA IDL

This approach takes advantage of the IDL types currently supported by existing CORBA implementations. It maps an XML document into CORBA types before an operation is invoked, and converts the CORBA types back to XML when they're received by the receiving object. This approach is network efficient and type-safe. Another advantage – the semantics of the document are preserved when translated directly into the CORBA structures used to encapsulate the document.

The examples that follow demonstrate the mapping between an XML document and its associated CORBA IDL definition. This mapping could be performed automatically given a DTD (or XML schema) for the XML document in question.

XML documents are implemented as a hierarchical CORBA type using a two-way mapping between XML and IDL and the CORBA Any. In this case, the CORBA Any will always hold a CORBA structure, and the contents of this structure depend on the XML document.

The mapping is detailed as follows:

MAPPING XML —> IDL

- An XML element maps to a CORBA struct and an appropriate primitive type is used for the element value:

```
<element>
    1.23
</element>
```

```
struct element
{
    float _Value;
};
```

- An XML attribute maps to a CORBA string member of the struct:

```
<element color='green' font='fixed' />
struct element
{
```

Object Management Group

www.omg.org

“While it’s possible to achieve integration using a primitive approach – converting the XML document to a string and passing this string to the object – little efficiency is achieved this way”

```
    string color,
    string font
};

<element color='green'>
    1.23
</element>
struct element
{
    string color,
    float _Value;
};
```

- A single child XML element maps to a struct within the parent struct:

```
<element>
    <child id='abc' />
</element>
```

```
struct element
{
    struct child
    {
        string id;
    };
};
```

- A variable number of XML child elements of the same name map to a sequence of structs within a struct:

```
<elements>
    <element color='green' />
    <element color='blue' />
</elements>

struct elements
{
    struct element
    {
        string color;
    };
    sequence<element> _element;
};
```

- A fixed number of child elements of the same name map to an array of structs within a struct:

```
<elements>
    <element color='green' />
    <element color='blue' />
</elements>
struct element
{
    string color;
};
struct elements
{
    element _element[2];
};
```

- A CORBA Type Code is produced to describe the newly defined structure and forms part of the resulting CORBA Any.
- An object that uses an XML document can be defined in IDL as follows:

```
interface MyObject
{
    void invoke(Any document);
};
```

MAPPING IDL → XML

- Only struct, sequences, arrays and base-types are mapped.
- An XML Schema or DTD is produced from the CORBA Type Code.
- A structs member maps to all attributes except “_ Value”, which maps to the text value for element.
- Sequences and arrays map to multiple elements.

This type-safe and efficient approach takes advantage of the available CORBA implementations. An example of mapping XML to CORBA taken from the finance domain demonstrates the mapping of a stock portfolio XML document to a CORBA IDL (see Listing 2).

Summary

In this article we discussed the benefits of using CORBA and XML for building a distributed application and three approaches to integrating XML documents with CORBA. While it’s possible to achieve integration using a primitive approach – converting the XML document to a string and passing this string to the object – little efficiency is achieved this way and the power of CORBA isn’t leveraged.

The second approach, which attempts to leverage from the standard DOM API to XML, presents a solution using the newly defined CORBA value-type specification. While this solution is efficient and type-safe, it doesn’t attempt to map the semantics of the document into the CORBA value type.

The final approach achieves efficiency and type safety, and succeeds in preserving the semantics of the XML document in the CORBA structure. It’s also possible to implement this approach with the currently available CORBA implementations. 🌀

References

1. W3C Web site: www.w3c.org
2. OMG Web site: www.omg.org
3. Hemming, M., and Vinoski, S. (1999). *Advanced CORBA Programming with C++*. Addison Wesley.
4. Slama, D., Garbis, J., and Russell, P. (1999). *Enterprise CORBA*. Prentice Hall.

AUTHOR BIOS

Nick Simha is the Western region presales manager for Iona Technologies (www.iona.com). He holds a master’s degree in computer science from the University of Missouri.

Dermot Russell is a senior XML software architect at Macalla Software (www.macalla.com), a Dublin-based software developer. Dermot has a BS in applied computing and an MS in computer science.

NSIMHA@IONA.COM DERMOT.RUSSELL@MACALLA.COM

XMLLeadership

www.brainstorm-group.com

LISTING 1

```
// module XMLValue
valuetype Node
{
    // NodeType
    const unsigned short ELEMENT_NODE = 1;
    const unsigned short ATTRIBUTE_NODE = 2;
    const unsigned short TEXT_NODE = 3;
    const unsigned short CDATA_SECTION_NODE = 4;
    const unsigned short ENTITY_REFERENCE_NODE = 5;
    const unsigned short ENTITY_NODE = 6;
    const unsigned short PROCESSING_INSTRUCTION_NODE = 7;
    const unsigned short COMMENT_NODE = 8;
    const unsigned short DOCUMENT_NODE = 9;
    const unsigned short DOCUMENT_TYPE_NODE = 10;
    const unsigned short DOCUMENT_FRAGMENT_NODE = 11;
    const unsigned short NOTATION_NODE = 12;

    private DOMString nodeName;
    private DOMString nodeValue;
    private unsigned short NodeType;
    private Node parentNode;
    private NodeList childNodes;
    private Node firstChild;
    private Node lastChild;
    private Node previousSibling;
    private Node nextSibling;
    private NamedNodeMap public;

    // Modified in DOM Level 2:
    private Document ownerDocument;

    Node insertBefore(in Node newChild, in Node refChild)
        raises(DOMException);
    Node replaceChild(in Node newChild, in Node refChild)
        raises(DOMException);
    Node removeChild(in Node oldChild) raises(DOMException);
    Node appendChild(in Node newChild) raises(DOMException);
    boolean hasChildNodes();
    Node cloneNode(in boolean deep);

    // Introduced in DOM Level 2
    // Modified for XML over CORBA from 'supports' to 'DOMsupports'
    boolean DOMsupports(in DOMString feature, in DOMString version );

    // Introduced in DOM Level 2
    private DOMString namespaceURI;
    private DOMString prefix;
    private DOMString localName;

    // Introduced for XML Over CORBA
    sequence<Node> children;
};
```

LISTING 2

```
<?xml version="1.0"?>

<Portfolio>

    <Stock name='TOI' url='http://www.toi.com'>

        <Holding>

            <Quantity>1000</Quantity>

            <Price>45</Price>

            <CurrentPrice>43</CurrentPrice>

            <Value>43000</Value>
```

```
<Performance>-2000</Performance>

    <Holding>

</Stock>

    <Stock name='PPP' url='http://www.ppp.com'>

        <Holding>

            <Quantity>3000</Quantity>

            <Price>25</Price>

            <CurrentPrice>30</CurrentPrice>

            <Value>90000</Value>

            <Performance>+15000</Performance>

        <Holding>

    </Stock>
</Portfolio>

struct Quantity
{
    float _Value;
};

struct Price
{
    float _Value;
};

struct CurrentPrice
{
    float _Value;
};

struct Value
{
    float _Value;
};

struct Performance
{
    float _Value;
};

struct Holding
{
    Quantity quantity;
    Price price;
    CurrentPrice currentPrice;
    Performance performance;
};

struct Stock
{
    string name;
    string url;
    sequence<Holding> holdings;
};

struct Portfolio
{
    sequence<Stock> stocks;
};
```



Evergreen

www.evergreen.com

BUILDING THE NEW

JavaCON 2000

The logo graphic for JavaCON 2000 consists of two square boxes. The left box is white and contains a stylized diamond shape split vertically into yellow and blue. The right box is black and contains a stylized 'J' and 'C' in white and blue.

CONFERENCE:
September 24-27, 2000

EXHIBITION:
September 25-26, 2000

**Santa Clara Convention Center
Santa Clara, CA**

Join 100% Java Developers


More than 2,500 of your peers will be at JavaCon 2000, along with the industry's most respected technical experts, sought-after gurus and advanced users, who will show you how to maximize Java for the enterprise. Make 2000 your year! Dedicate yourself to four days of the hottest Java techniques thought by those who are defining Java's future.

www.JavaCon2000.com

Presented by:  **SYS-CON
MEDIA**

 **JAVA
DEVELOPERS
JOURNAL**

Produced by:

 **CAMELOT**

Register
by **AUG. 18**
and
SAVE \$200!



Java Developer's Journal Puts a Twist on Today's Hottest Topics

JavaCon 2000 is your only opportunity to

learn from the experts at *Java*

Developer's Journal –

who best combine
technical expertise
and practical
vendor know-how.



BENEFITS OF ATTENDING

- The tips and techniques you'll learn will help you do your job better.
- Discover the new applications being developed today that you'll need tomorrow.
- Sessions are designed for users at all levels, with special sessions just for gurus.
- Network with fellow software developers as well as recognized Java experts.
- You'll learn how Java is being used for large-scale enterprise applications.

TWO-DAY EXHIBIT

The full-scale exhibit hall will be packed with leading vendors who will be on hand to demonstrate the latest products and answer your questions.

Exhibit Hours:

Monday, September 25,

12:00–6:30

Tuesday, September

26, 12:00–6:00

Every delegate receives
a **FREE** one year
subscription to
XML-Journal and a one
year free subscription to
Java Developer's Journal—
a **\$99.00** value

ARE YOU A JAVA GURU?

Why Not Join Our Faculty?

If you're willing to share your unique Java experience, please e-mail stewart@camelot-com.com for details on how you may join the faculty. Topics of interest include but are not restricted to:

- Java in the Enterprise
- Embedded Java
- Java Success Stories
- Advanced Java Application Development
- XML & Java
- Java in the Industry/Java Business Applications
- Server-Side Java
- Java Testing and Debugging
- Object Oriented-Concepts and Design with Java

Deadline: April 17, 2000

Acceptance: May 15, 2000

JAVA DEVCON 2000 TECHNICAL CHAIR



Joshua Duhl is a principal at Stillpoint Consulting. For over 12 years he has worked with companies using object technologies including object databases, application servers and object development tools or with companies developing products for the Internet and Web. An industry analyst associated with IDC covering object technologies and emerging markets, Duhl was one of the five original authors of the ODMG-93 ODBMS standard.

Take a Look at What You'll Learn:

- Building Mission-Critical Applications with Java
- Advanced JFC/SWING Component Integration
- Real-Time Java
- Designing High Performance E-Commerce Systems with EJB
- Combining XML and Java
- Java 2 Platform, Enterprise Edition
- Java Advanced Programming
- Java Security
- Java Runtime Internals
- Building a Multithreaded Server in Java
- Methods for Effective Java Unit Testing
- Developing COM and MTS Components in Java
- Developing Multi-tier Applications Using the Servlet API
- Object-Oriented Analysis and Design with Java
- Dynamic Bytecode Generation with Java
- Developing Large-Scale Applications with Java and CORBA
- JDBC Technology
- 2D and 3D Graphics in Java
- Designing Java Business Applications
- Java and Legacy Systems
- Using the Java Naming and Directory Interface API
- Java Gaming
- Programming for Devices (J2EE)
- Programming for the Desktop (J2SE)
- Java Commerce
- Using Java Agents
- Jini and JavaSpaces
- Java Testing and Debugging
- Java Exception Handling
- Garbage Collection Techniques
- Using Java with UML
- Writing Consumer Applications Using Personal Java API
- Smart Card Application Development
- Database Integration with Java



The extensibility of XML is revolutionizing e-business

XML: It's the 'X' that Matters

As XML survives its debutante ball and begins to be accepted by mainstream IT shops, it's being put to work, creating excitement among CIOs with its extensibility. Having had first-hand experience with several next-generation XML e-business application deployments, I'd like to describe how the extensibility of XML is revolutionizing e-business, making it possible to finally develop applications that are flexible enough to keep pace with today's constantly changing business requirements.

What Is Extensibility?

When we talk about XML's extensibility, we're talking about the ability to dynamically create new tags or attributes to a single record. For example, consider a database for an inventory of cars in which each car has a make, model and price. If you decide to add a used car to your inventory, you may also want to record the mileage. XML's extensibility enables you to create a new record for that car with the additional attribute without disrupting any of the other records.

Even more compelling is how easy it is to include ad hoc, unstructured data to an XML record. Since you can create a new tag and extend a record at any time, you can record a dent by adding a <damage> attribute. You can even add your daughter's auto-savvy witticisms with an <alex> tag. The ability to easily incorporate unstructured data is at the heart of e-business because then you can begin to leverage all the information that's not in a formal data-management system. What percentage of information on your hard drive or in your head can be searched, cross-referenced or shared with partners?

This isn't possible with traditional technology. In order to add a new attribute to a record in a relational database, you'd have to add a new column to the entire table. This may be acceptable if there are relatively few additional attributes, but if every new business partner (such as a car dealership) or new

product configuration (such as a new vehicle type) demands different attributes, it quickly becomes an unwieldy way to manage your e-business application.

The extensibility of XML also passes the interoperability test. Since XML is portable and an industry standard, it's the de facto format for data sharing between applications over the Web and across organizations. When an XML data object is passed from one server to another, the ASCII tree is parsed and the application can extract the elements and attributes it needs. If there are new and unknown attributes, they're simply ignored. Compare this to what would happen if a C++ or Java object contained unexpected fields – you'd end up with skewed data and memory leaks.

The extensibility of XML enables you to add new fields and attributes to data records at will, without fear of disrupting any other records or breaking applications. This is why XML is not just another "EDI" and why it will succeed where EDI failed. XML's extensibility is causing developers to think differently about system design and to be less fearful of the rapid changes that occur with e-business.

E-Business Demands XML Extensibility

The Gartner Group defines the role of e-business as follows: "E-business enables and manages relationships between an enterprise and its functions and process-

es, and those of its customers, suppliers, value chain, community and industry." Successfully implementing e-business applications requires leveraging information assets from many different sources and in many shapes and sizes – structured and unstructured, familiar and proprietary. To complicate matters, this information needs to be consumed by many types of applications and users. XML is the ideal solution to this "Tower of Babel" because it's simple, flexible and portable. It's the standard for e-business.

E-business applications are a new breed of application. They require constant change because corporations need to deliver targeted products and services to their customers faster than the competition can. Organizations need to work with partners over an extranet as easily as with people down the hall. E-business applications have to be able to withstand new products, services and partners without requiring rearchitecting. XML's extensibility is the key to making this possible.

XML Makes Customization a Breeze

Because of XML's extensibility, applications can be customized to accommodate new requirements without any code having to be rewritten. XML applications tend to be more like frameworks that are dynamically customized by the data model. In the car example above, an application might list all the attributes of a given vehicle and let you query on any of them, but the car's description and the search menu wouldn't be generated until runtime. If you wanted to add trucks to your inventory, and now towing capacity is important, simply extend the data record with

AUTHOR BIO

Coco Jaenicke is the XML evangelist and product marketing manager for eXcelon Corporation. She's played a key role in the successful development and introduction of eXcelon, the industry's first XML application development environment for building and deploying dynamic e-business applications.

that new attribute and the application takes care of itself.

This concept can be applied to a variety of different e-business application areas.

- **On-line product catalogs or e-commerce sites** are constantly changing with new products or new departments. By extending XML with new information, such as customer comments or an auto-loan worksheet, the Web site can accommodate changes without redesigning data structures or destabilizing existing applications.

- **Business-to-business (B2B) extranets** can reap the same benefits. Many times when businesses come together they won't be using the same semantic expressions to describe the same information (for example, one vendor may use "price" where another uses "cost"). The various competing DTDs already available virtually assure this. Because of the portability of XML, the syntactical problem of sharing data is separated from the semantic problem, and because of its extensibility, differences in semantics can be accommodated as well. If a partner supplies you with information about a car with a "price," you can extend the data object to add a "cost" attribute and both your application and your partner's will continue to operate safely.

- **Enterprise Application Integration (EAI)** is much simpler with XML. With EAI, adding or upgrading a module can have a ripple effect that destabilizes any connected system. For this reason upgrades often have to be performed across an organization in unison. For example, new middleware has to be installed on every server, much to the chagrin of every MIS director. XML lets you extend data objects to accommodate new modules without breaking the old ones, so you can upgrade servers asynchronously. This means that if your growing automobile enterprise gets a new accounting system that requires you to track the date of every transaction, you can add that attribute to every car without worrying about disrupting your Web-based showroom.

Conclusion

XML has received inordinate amounts of praise and hype, most of it coming from the fact that it's portable, standard and – let's face it – just plain fun to use. XML has survived the initial skepticism and is emerging as an essential tool for e-business primarily because of its extensibility. The extensibility enables applications to turn on a dime, leverage structured and unstructured data, and take many of the major headaches and fears out of managing a major MIS initiative. What more could you ask for? ☺

COCO@EXCELONCORP.COM

Meet JDJ EDITORS AND COLUMNISTS

Attend the biggest Java developer event of the year and also get a chance to meet *JDJ's* editors and columnists

JavaCON 2000

MEETING
September 24-27, 2000

Santa Clara Convention Center
Santa Clara, CA

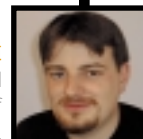
Sean Rhody Editor-in-Chief, *JDJ*

Sean is the editor-in-chief of *Java Developer's Journal*. He is also a principal consultant with Computer Sciences Corporation.



Alan Williamson *JDJ*/Straight Talking Columnist

Alan, the Straight Talking columnist of *JDJ*, is a well-known Java expert and author of two Java books. A contributor to the Servlet API. Alan is the CEO of n-ary Consulting Ltd, with offices in Scotland, England and Australia.



Ajit Sagar Editor-in-Chief, *XML-Journal*

Ajit is the founding editor of *XML-Journal* and a well-respected expert in Internet technologies. A Sun-certified Java programmer, he focuses on Web-based e-commerce applications and architectures.



Jason Westra EJB Home Columnist

Jason is the Enterprise JavaBeans columnist of *JDJ* and a managing partner with Verge Technologies Group, Inc., a Java consulting firm specializing in Enterprise JavaBeans solutions.



JAVA DEVELOPER'S JOURNAL

CAMELOT

ADVERTISING INDEX

ADVERTISER	URL	PH	PG
ACTIVATED INTELLIGENCE	WWW.ACTIVATED.COM	919.678.0300	67
ELIXIR TECHNOLOGY	WWW.ELIXIRTECH.COM	65 532.4300	33
EVERGREEN	WWW.EVERGREEN.COM	480.926.4500	49
EXCELEON	WWW.ODI.COM/EXCELEON	800.706.2509	34,35
IBM	WWW.IBM.COM/DEVELOPERWORKS	800.772.2227	68
JAVACON 2000	WWW.JAVACON2000.COM	212.251.0006	50,51
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	54,55
MICROSOFT	WWW.MSDN.MICROSOFT.COM/EVENTS		59
MICROSOFT	WWW.MSDN.MICROSOFT.COM/TRAINING		61
MICROSOFT	WWW.MSDN.MICROSOFT.COM		63
OBJECT MANAGEMENT GROUP	WWW.OMG.ORG	781.444.0404	45
SD 2000	WWW.SDEXPO.COM	800.441.8826	39
SEQUOIA SOFTWARE	WWW.XMLINDEX.COM	888.820.7917	13
SILVERSTREAM	WWW.SILVERSTREAM.COM	888.823.9700	27
SIMPLEX KNOWLEDGE CO.	WWW.SK.COM	914.620.3700	65
SOFTQUAD	WWW.SOFTQUAD.COM	416.544.9000	2
SOFTWARE AG	WWW.SOFTWAREAG.COM/TAMINO	925.472.4900	15
TANGO DEVELOPER'S JOURNAL	WWW.TANGOJOURNAL.COM	800.513.7111	7
VSI	WWW.VSI.COM/BREEZE	800.556.4VSI	21
XML DEVCON	WWW.XMLDEVCON2000.COM	212.251.0006	8,9
XML LEADERSHIP	WWW.BRAINSTORM-GROUP.COM	508.393.3266	46

JDJ Stor

www.jdjs

e Spread

store.com



XML is rapidly turning all other integration approaches into historical curiosities

Business-to-Business E-Commerce: XML's Killer App

Every technology we accept as standard and ubiquitous – from PCs to the World Wide Web – has achieved that level of overwhelming acceptance because of a “killer app” or other enabling technologies. For PCs it was spreadsheets. For servers it was the relational database. The Internet might have remained a collection of academic bulletin boards if it hadn't been for its two killer apps: e-mail and the World Wide Web.

In each of these cases an application emerged that exploited the potential of the new technology to fulfill an unmet need, powerfully and easily. For XML, e-commerce is proving to be the killer application.

Enterprise Business Integration and the Emergence of XML

A fast-emerging driver for XML adoption has been Enterprise Application Integration (EAI) or, more accurately, Enterprise Business Integration (EBI) – the integration of data, applications and processes across multiple functions in the enterprise and beyond.

Before XML, companies had a difficult choice. They could either hard-code point-to-point integration solutions, creating the proverbial spaghetti, or they could adopt a proprietary integration solution and, with it, a proprietary canonical data representation. Either one restricted future options and created an ongoing maintenance burden. Through its openness, simplicity and industry-wide support, XML provides a sound and flexible underpinning for integration solutions.

EAI has already raised the profile of XML considerably. But many companies have lacked the will or the urgency to pursue a systematic approach to integration – and, consequently, to adopt XML – until something came along to give the business a huge push: e-commerce.

Creating a Well-Differentiated E-Commerce Strategy

For many companies initial e-commerce implementations typically focused on providing Web-based product information with a secure order-entry module. While these early efforts have allowed companies to gain a toehold in e-commerce, they were inefficient and ineffective since they were disconnected from the operational systems of the business.

Creating a well-differentiated e-commerce strategy requires the ability to provide unified information to a range of users such as employees, customers, suppliers and partners. Organizations must integrate customer-facing components with front- and back-office applications, as well as with legacy systems. Furthermore, they need to create and deploy integrated business processes that differentiate the business and add real value for the customer.

Those very demanding requirements are certainly far beyond the capabilities of any point-to-point integration solution. To put it simply, you can't do business-to-business e-commerce effectively without EAI. And as the world rushes to XML, you won't be able to do EAI effectively without XML.

The Role of XML in Business Integration

XML fills a vital role in business integration by providing a generalized mechanism for representing and structuring information. XML is in fact a

“metalanguage,” which means it can be used to define any set of constructs and is hence inherently extensible. As a result, XML provides not only a replacement for HTML but also a flexible framework for representing the structured data associated with databases and application systems. Any data structure can be rendered as an XML document.

Just as HTTP has become the standard transport protocol for Internet computing, XML is rapidly becoming the standard for data exchange. In its earliest applications XML provided a “more powerful HTML” for interfacing structured data with Web-based applications. More generally, it's also emerging as a flexible vehicle for storing, manipulating and exchanging data of all types across organizations, systems and technologies.

The power of XML lies in its ability to represent the data itself and to define its structure and meaning. XML relies on extensible text tags (or elements) to describe data structures and formats. Using XML, an organization can specify a vocabulary of data elements in, say, a customer-processing application such as the name, street address, city/state/zip, phone number and customer number. Different applications can then identify that data, interpret its attributes and then use it appropriately.

DTDs and Schemas

Over the years there have been many initiatives to define standard data representations to facilitate integration between systems and organizations. The more successful have included standards

for EDI, the HL7 standard within health care and interbank settlement systems. However, such standards have generally offered limited flexibility, suffered from multiple dialects and been applied only within their narrow domains. As the requirements for a broader approach to information sharing and application integration have emerged, these technologies lack the required generality, simplicity and flexibility. Today the focus for such initiatives has shifted wholeheartedly to XML – indeed, all these historical standards are in the process of being redefined within the XML framework.

The emergence of vertical and horizontal schemas (and DTDs) is what truly facilitates the use of XML for EAI. The first solutions came out of the academic community and covered such areas as chemical structures, mathematics and data documentation for social science (DDI). Many industry trade groups, vendors and consortiums are now defining schemas for their particular industry or areas of special focus. Schemas become valuable to EAI when they provide a standard for vertical markets, such as financial settlements or telephone billing, or a more generalized business function, such as credit verification.

The use of common schemas becomes especially compelling when forging integration with another organization's applications, as in business-to-business (B2B) solutions. These industry agreements eliminate the need for organizations to hammer out their own definitions and secure agreements between each of the individual parties. They also provide a common specification to be adopted by applications' package and service providers.

XML-Enabling Applications

Until recently, packaged applications have imposed largely proprietary interfaces. As a result, traditional integration solutions have required custom connectors to deal with each application's API. Furthermore, API-based integration requires a common contract in terms of middleware (CORBA, COM, DCE, etc.) that in turn creates dependencies and tighter coupling between the systems. On its own, XML may simply be seen as a standardized data representation format; when coupled with HTTP, XML becomes a ubiquitous middleware solution that lends itself to the loosely coupled style of integration required by EAI solutions. In addition, XML is increasingly being supported by other message transports such as JMQ and MQSeries.

In response to the fast-growing demand for interoperability, especially over the Web, major application vendors, including SAP, Oracle and Siebel, are now rushing to add XML-based APIs to their application suites. Such initiatives eliminate the need for custom connectors for these packages.

In addition, a third-party market for XML-based connectors for popular applications is rapidly emerging. For example, as part of its recently announced Open Integration Framework initiative, PeopleSoft will deliver XML-based APIs that enable developers to plug into PeopleSoft business processes without requiring detailed knowledge of the underlying data structures.

For legacy applications, custom wrappers must be provided to deal with the native APIs and to convert the native data streams to an XML equivalent. Given the prevalence of legacy and custom applications, the ability to create new connectors rapidly is a critical factor in the success of integration solutions.

Forrester Research forecasts that application providers will soon bundle XML translators into their products. These translators will support XML and the industry-specific schemas for their target markets. Wherever a set of applications supports a common schema, the need to provide custom data integration and transformation services will be removed.

XSL as a Data Transformation Mechanism

For broader enterprise solutions, however, a requirement will remain for data transformation to support legacy formats and to provide interoperability across schemas. For example, a single organization may support an EDI schema within its supply chain but a totally different schema in its manufacturing systems. A key benefit of XML is that it comes fully equipped with a native data transformation mechanism, XSL (eXtensible Stylesheet Language).

Applying an XSL stylesheet to an XML document produces an output document (typically XML or HTML) transformed by the application of the relevant XSL rules. The stylesheet concept provides a clean separation between the content itself and the specific format required by a target application or output document. By separating the definition of content from the format in which it's used, XML makes it possible to share information across multiple requirements.

In early draft specifications XSL primarily provided a mechanism for manipulating tags, particularly to allow specific formatting elements to be applied for presentation purposes. Today XSL provides not just a formatting capability but also, through XSLT, a full transformation capability able to manipulate both tags and data content.

The effect is that XSL has emerged as a standards-based data transformation capability for XML-based data and the ideal vehicle for the data manipulation aspects of e-commerce.

In traditional integration projects, transformation has been custom-coded or implemented through a proprietary data transformation tool. The advantages of employing XSL for transformation lie in the clear separation of transformation rules from the application programming effort and in its seamless integration with XML.


XSL becomes the natural way to provide schema-to-schema transformation in the XML world. In the future, XML applications' initiatives will not only define the schemas themselves but also specify transformation templates in XSL to make their data readily available to other applications' domains.

Conclusion

XML – THE LANGUAGE OF E-COMMERCE

XML promises to achieve for structured information what HTML achieved for text and graphics on the Web.

- XML is rapidly emerging as the preferred data integration backbone within and across organizations and industries.
- XSL provides a built-in mechanism for dealing with different data semantics across applications and domains.
- XML, in conjunction with XSL and HTTP, provides for the customized delivery of information to the browser, a prerequisite for compelling customer-oriented applications.

Business-to-business e-commerce is fueling the adoption of XML. The Internet has rewritten the rules for supply chain management, redefined telephony, set new standards for 24-hour customer service and spawned new business models. These new Web-based systems must be effectively integrated with applications from partners, suppliers and external service providers such as credit card vendors and shippers. XML is the lifeblood of this new world, and is rapidly turning all other integration approaches into historical curiosities. 

AUTHOR BIO

John Spiers is vice president of international marketing and Internet application and performance tools at Sun Microsystems. A well-known figure in the IT industry, John frequently appears in the press and on speaking platforms. He holds a master of arts degree from Cambridge University.

 JSPIERS@FORTE.COM



An open-interchange model that brings consistency and compatibility to applications

The Specification for the Metadata Interchange Format

It's finally here – the specification for Components' Metadata Interchange based on XML. XML Metadata Interchange (XMI) format specifies an open-interchange model intended to give developers working with object and distributed technology the ability to exchange data between tools, applications, repositories, business objects and programs. This is a stream-based model interchange format (SMIF) that enables the exchange of modeling, repository and programming data over the network and Internet in a standardized way. XMI is a much-needed specification to bring consistency and compatibility to applications created in collaborative environments.

This article focuses on the details of the technology, its benefits and its architecture, and how it fits into the Object Management Group's (OMG) modeling and repository architecture. We provide a simple example to illustrate the technology that was waiting for OMG's technical vote as of March 1999.

Technology in a Snapshot

XMI is an open, stream-based interchange format formed by the integration of three key industry standards:

- **XMI:** A W3C standard that defines an open, metamodel-neutral, programming language-neutral, API-neutral, streamable, textual, human-readable format to bring structured information to the Web
- **UML:** Unified Modeling Language, an OMG modeling standard that defines a rich, object-oriented modeling language/notations for object-oriented analysis and design (OA&D)
- **MOF:** Meta Object Facility, an OMG metamodeling and metadata repository standard that specifies an extensible framework for defining models for metadata and represents it as CORBA objects; uses UML notations for models

The origin of XMI can be traced to November 1997, when MOF and UML were adopted as OMG standards. However, because of lack of time, SMIF was not specified. Thus, in December 1997, OMG issued an RFP for SMIF. This received three initial submissions – XMI, CDIF and UOL – that are now integrated into one, XMI. Figure 1 shows the position of SMIF in the OMG repository and modeling architecture.

Why an XML-Based Interchange?

XML, which is based on SGML (Standard Generalized Markup Language), is a simple, flexible, tagged format designed for information interchange. The key feature, which enables XML to be chosen over other formats, is its ability to separate data and metadata, presentation and content. This architecture of contextual separation enables generic tools (like XML parsers) to validate an XML document against its grammar, commonly known as *data type definition* (DTD).

Thus XML-based metadata interchange boils down to defining DTDs for MOF, UML and other standards. With this power it isn't surprising to see that the support for XMI is industry-wide.

The key aspects of the architecture are:

1. A four-layered metamodeling architecture for general-purpose manipulation of metadata in distributed object repositories.
2. The MOF model is used as the metamodel, while MOF is used to define and manipulate metamodels programmatically using fine-grained CORBA interface. This approach leverages the strength of CORBA distributed object infrastructure.
3. UML notation is adopted for representing models and metamodels and to describe the semantics of OA&D models.
4. XML enables SMIF.

Figure 2 shows XMI in a nutshell and Figure 3 shows the OMG's four-layered metadata/metamodeling architecture. The

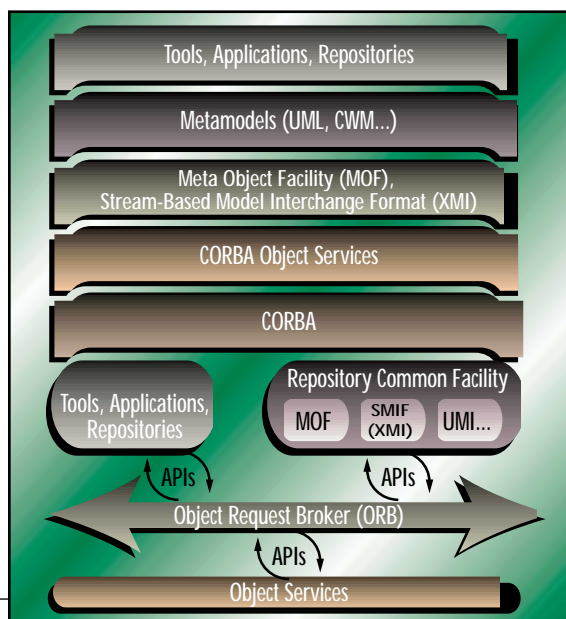


FIGURE 1 OMG repository and modeling architecture

Microsoft

www.msdn.microsoft.com/events

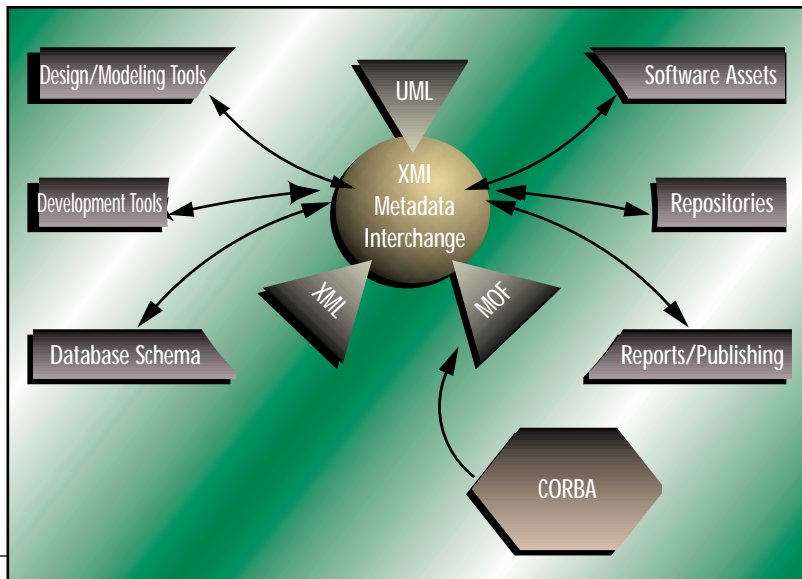


FIGURE 2 XMI in a nutshell

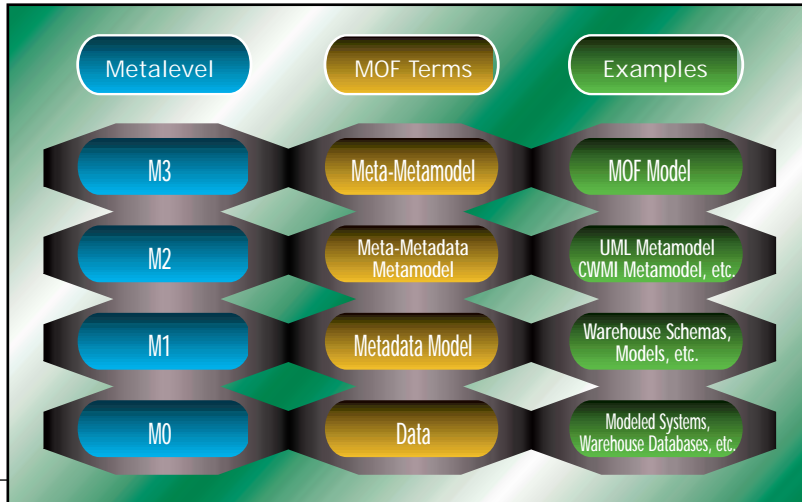


FIGURE 3 OMG's metadata/metamodeling architecture



FIGURE 4 A simple example of an XML document fragment

current XMI architecture is now extended to include data warehouse metadata (Common Warehouse Metadata Interchange RFP) and other metadata by defining MOF-compliant metamodels.

What the Specification Provides

The current specification focuses on import and export of metamodels and models including extensions. As stated above, the current architecture is being extended for interchange of repository, data-warehouse data.

The current specification provides an unambiguous XML DTD for UML-based models (UML DTD) as well as MOF-based metamodels and their instances (MOF DTD). It also provides a precise MOF-to-XML mapping that allows interchange of any MOF-based metamodel and corresponding models (MOF→XML Stream) and enables automatic generation of DTDs for any MOF-based metamodel (MOF→XML DTD). In addition, it recommends the use of UML and MOF for metamodel design.

Figure 4 shows an example of a simple model and its XML document fragment.

Conclusion

Adoption of this technology opens up many benefits to the distributed object industry. For example:

- It enables open interchange of the MOF meta-metamodel, UML and other information assets between vendor tools (this helps save time on evaluating and choosing a specific modeling/repository tool like Rational Rose, System Architect, Select, TeamConnection, Oracle, etc.).
- It leverages the existing XML/HTML infrastructure to publish design metadata on the Web.
- It ensures compatibility between application tools, IDEs, languages, databases and the like in heterogeneous environments.
- It facilitates exchange of metamodel information over the Internet for developers working in a remote, distributed or intermittently connected environment.
- It provides middleware-neutral open interchange, which will be of significant benefit in non-CORBA and hybrid CORBA environments.

This exciting, desirable technology advances our quest for open standards in distributed heterogeneous environments. 🌐

AUTHOR BIO

Ilango Kumaran S, an associate at The Technical Resource Connection, Inc., based in Tampa, Florida, has worked on the design and development of distributed object and object systems for more than 10 years. He has extensive experience with Java, EJB, RMI, CORBA, C++, RDBMS, OODBMS and Web technologies. Ilango, who holds an MBA from IGNOU, India, and an MS in engineering from Anna University in India, has written extensively about emerging technologies.

Microsoft

www.msdn.microsoft.com/training

XML NEWS

SoftQuad Software, Software AG Form Strategic Alliance

(Toronto, ON, and Darmstadt, Germany) – SoftQuad Software Inc. and Software AG have

announced a joint initiative to integrate SoftQuad's XMetaL XML authoring environment and Software AG's Tamino XML Information Server. The integrated products provide customers with a complete solution for developing, managing and publishing XML content for a wide variety of electronic business applications.

www.softquad.com 



IBM Selects XML.ORG

(Boston, MA) – OASIS, the vendor-neutral organization for XML interoperability, announced that IBM has submitted its Trading Partner Agreement Markup Language (tpaML) for standardization within the OASIS XML.ORG initiative. Developed by IBM, the tpaML specification uses XML to define and implement electronic contracts.

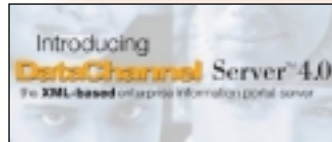
The foundation of tpaML is the Trading Partner Agreement (TPA), which defines how trading partners will interact at the transport, document exchange and business protocol layers. XML-based TPA documents capture the essential information upon which trading partners must agree in order for their applications and business processes to communicate.

www.oasis-open.org 



DataChannel Server 4.0 Released

(Bellevue, WA) – DataChannel has released its flagship XML-based Enterprise Information Portal Server, DataChannel Server 4.0



Enterprise Information Portal Solutions from DataChannel provide customers with a personalized e-business interface to mission-critical information. This information can be dynamically categorized into "channels" that are then published in a personalized view to the right user, inside or outside the enterprise.

www.datachannel.com 

Object Design Changes Name; Announces New Products

(Burlington, MA) – Object Design, Inc., has changed its name to eXcelon Corporation and announced that its new NASDAQ ticker symbol is EXLN. The name change reflects eXcelon Corporation's increasing role in the expanding market for business-to-business solutions.



The company is also extending its product family to include a new B2B integration server aimed specifically at the B2B solutions market, and is announcing the eSolutions services organization that will provide industry-specific e-business frameworks and solutions.

www.exceloncorp.com 

Microsoft Announces Support for XSLT and XPath

(Redmond, WA) – Microsoft has released a preview version of its XML parser, delivering on its commitment to support the W3C's latest recommendations of XSL Transformations (XSLTs) and

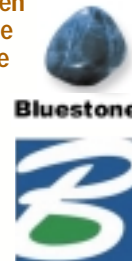


XML Path Language (XPath). The parser, which includes new features that increase performance and efficiency, is available for immediate download at

<http://msdn.microsoft.com/downloads/webtechnology/xml/msxml.asp>. 

Bluestone and Bentley to Bring XML-Driven Applications to the A/E/C Marketplace

(Houston, TX) – Bluestone Software, Inc., and Bentley Systems, Inc., are bringing XML-driven engineering project application services to the \$3.6 trillion architecture, engineering and construction (A/E/C) industry.




The two companies have entered a three-year agreement that gives Bentley license to develop engineering software solutions based on Bluestone's award-winning Sapphire/Web Application Server, Bluestone XML Suite Integration Server and Bluestone's comprehensive standards-based, e-business solution, Total-e-Business.

www.bentley.com
www.bluestone.com 

OAGI Releases 122 XML Based Business Messages

(Chicago, IL) – The Open Applications Group, Inc. (OAGI), the largest publisher of XML-based business messages in the world, announces the publication of their latest specifications for e-business and application integration. Release 6.2 of the busi-




ness process-based Open Applications Group Integration Specification (OAGIS) contains the largest and richest set of XML DTD files in the world and goes farthest toward defining the "digital dial tone" that organizations require to do business in the emerging e-world. These specifications may be downloaded for free at www.openapplications.org. 

SYS-CON Publications, XML-Journal Announce XML DevCon 2000

(Pearl River, NY) – XML-Journal is launching a key, XML-focused conference and expo scheduled for June 25–28 at the New York Hilton in New York City. XML DevCon 2000, cosponsored by Oasis and produced by Camelot Communications Corp., will feature an intense four-day technical program focused on how to maximize XML for the enterprise. Delegates will have the opportunity to master new skills from an exceptional lineup of educational tracks designed for users of all levels.



"XML DevCon 2000 will provide valuable insight for XML developers charged with maximizing XML to build enterprise applications," said Fuat Kircaali, founder of SYS-CON Publications and publisher of XML-Journal. "As the only national XML event organized on the East Coast in 2000, XML DevCon 2000 will be a historic forum where XML and Internet technology professionals will be able to meet and share their experiences with fellow software developers and recognized industry experts."

For online registration visit www.XMLDevCon2000.com or call 212 251-0006. 

Microsoft

www.msdn.microsoft.com

XML NEWS

GA eXpress Initiates New Era of E-Business with Products Featuring XML

(Irvine, CA) – GA eXpress, formerly General Automation, has introduced products that will support its recently announced “ePath” strategy by enabling enterprise-wide e-business solutions for the \$3 billion MultiValue marketplace. Demonstrated for the first time at the Spectrum 2000 conference in Anaheim,



California, the suite of platform-independent connectivity and e-commerce middleware products give GA eXpress customers a path to build an industry-compliant foundation for migrating, manipulating, moving and gathering business-critical information electronically.

www.gaexpress.com

HR-XML Consortium Expands Membership, Initiates Standards and Workgroups

(Research Triangle Park, NC) – The HR-XML Consortium announces that more than 45 organizations have joined the consortium for the purpose of creating and promoting standardized, HR-specific XML vocabularies.



In addition, five members have been elected to the consortium's board of directors: Jeff Bonar (Ultimate Software), David Donahue (Aetna), Gary O'Neill (Icarian), Lon Pilot (Watson Wyatt) and Cari Willis (IBM). The board is tasked with leading the nonprofit group in creating and promoting a standardized XML framework for a broad range of human resource-related transactions and intercompany data exchanges.

www.hr-xml.org

OnDisplay Launches Industry's first Free B2B XML Server

(San Ramon, CA) – OnDisplay, Inc., a leading provider of e-business infrastructure software for powering e-business portals and e-marketplaces, will deliver the industry's first free business-to-business XML server software for any organization that needs to establish secure,

guaranteed exchange with online trading partners. Called XML Connect, the new product

enables the exchange of XML business documents – such as purchase orders, invoices and order con-

firmations – seamlessly and securely with any other XML Connect user, as well as with users of OnDisplay's CenterStage eBizXchange product.

XML Connect will be generally available on March 30 as a free download from the XML Connect Web site

(www.xmlconnect.net) and from OnDisplay's Web site

(www.ondisplay.com). The product will include free online support.



www.address.com

Internet Visionary Joins XMLSolutions

(McLean, VA) – XMLSolutions Corporation, the only provider of XML-based EDI Solutions and XML Schema Management products and services, has announced that Edwin Miller has joined the company as president and chief operating officer.



A leader in the Internet, communications and software industries since 1993, Miller has extensive understanding of new business models and the digital economy, including business-to-consumer e-commerce, ISPs, Web portals, interactive advertising, software development and publishing. www.xmls.com

OASIS and HL7 Exchange Memberships

(Boston, MA and Ann Arbor, MI) – In a move that is expected to further XML application and interoperability in the healthcare industry, OASIS and Health Level Seven (HL7) have exchanged sponsor memberships. The reciprocal membership opens communication between the cross-industry efforts of OASIS, which



advances XML interoperability through its XML.ORG industry portal, and the industry-vertical XML development work of HL7, which has made great strides in implementing XML for clinical patient and healthcare services. Both groups are international in their focus. www.hl7.org www.oasis-open.org

Introducing Stilo XMLDeveloper

(Cardiff, Wales) – UK e-commerce development tools and services specialist Stilo Technology announces a breakthrough in applications development with the launch of Stilo XMLDeveloper. This software delivers huge savings in development time, enabling organizations to bring new e-commerce applications to market faster than ever.



www.stilo.com

Information Architects Offers Industry's First XHTML Conversion Solutions

(Charlotte, NC) – Information Architects announces its support for the W3C Recommendation for XHTML 1.0.



Bridging HTML with the power of XML, XHTML provides flexible Web pages and robust Web applications for a wide range of platforms and browsers, including handheld computers, mobile phones, PDAs, two-way pagers, televisions and kiosks.

Offering scalable, bidirectional dynamic exchange of both structured and unstructured data, content and commerce functionality in any format, iA's Metaphoria products automatically transform static and dynamic documents to HTML, XHTML and advanced data formats such as XML and RDF. Further information on XHTML 1.0 can be found on the W3C Web site at

www.w3.org/

RSA and Netfish Introduce XML-Based Solution for Baan Customers

(Englewood, CO, and Santa Clara, CA) – RSA Companies and Netfish Technologies have formed a strategic alliance that now allows the two companies to deliver a set of Internet-enabled business process integration tools and solutions to the Baan customer base. The joint development effort has



resulted in a “Baan Adapter” that will seamlessly integrate the Netfish XDI system with the mission-critical applications within the suite of Baan BackOffice applications. The companies also announced that RSA Companies has become a certified sales and integration partner of Netfish and will be delivering Netfish products and services to customers of Baan as well as other ERP systems.

www.netfish.com

<http://rsacompanies.com>

Simplex Knowledge Co.

www.skc.com

Meaning, **Not** Markup



WRITTEN BY SIMON PHIPPS]

This article explores the paradox that sharing a common vocabulary can actually restrict the richness and nuances of a business paradigm.

The Trend to Share Common Vocabularies

One of the trends we find in the rapidly expanding world of XML is a desire to define unified vocabularies for expressing exchanged data. It seems obvious that the number of XML vocabularies used should be minimized, and, indeed, efforts to find agreement within various industries are laudable and to be encouraged. But these efforts may not always reduce the difficulty of exchanging information and might, paradoxically, limit the longevity of information by hiding complexities behind superficial agreements. The issue is meaning, not markup.

Consider this illustration. At a summit of religious leaders aimed at increasing common understanding among the world's religions, it is decided that everyone will speak English and use the vocabulary of Protestant Christianity. However, as soon as the discussions start, there are problems. Someone uses the word *heaven* and many people nod in recognition. But as the discussion progresses, it is clear that even the different Christian delegates have understood different nuances of the word, to say nothing of the Hindu and Buddhist representatives. As time goes by, they realize that perhaps they should have agreed at the start not to use a single vocabulary but rather to describe what the relationships were between the apparently similar words in the vocabularies with which they were already familiar.

In the same way, a shared vocabulary for users in the same domain may not be enough to allow them anything but the most rudimentary sharing of information. Different organizations usually have diverse backgrounds and varying views of the world, and their competitive advantages often result from their different paradigms. When all that is involved is basic data (to do with billing or ordering, for example), the issues may be trivial, but full-scale cooperation between companies will increasingly involve mind-to-mind connections. At this point, when differences come to light they will result from trying to cram paradigmatic variations into the same syntax. Resolving differences will be hard even when the difference has been detected because the semantic strength of the vocabulary in use may not allow for the proper expression of the alternate paradigm.

The Paradox

So we discover a paradox. It would seem that common purposes should share common vocabularies. Yet in defining the data model for a business, it may be better to invent a local variant of an existing vocabulary – or in extreme cases a whole new vocabulary – to fully express the richness and nuances of a business model and paradigm. This was, after all, the feature of XML that first drew the attention of both the publishing and the computing worlds – that magical letter X for eXtensible. The chief skills for the use of XML will prove to be those of the business analyst, defining data and naming parts, rather than those of the programmer.

The Need to Map the Relationships Between Paradigms

Translation will therefore be key. There is certain to be a need to move between different vocabularies, be they supersets of standards or custom vocabularies. In these early days of XML it will no doubt be enough to use straightforward transliteration functions. The functions of the proposed XSL standards are more than enough to allow exchange between basic business vocabularies. If the number of these vocabularies can be kept to a minimum, the task of setting up the simplest transactions could be easy, especially using visual tools to perform the mapping. But there will always be a mapping, even if there is only one common exchange vocabulary. The mapping describes the relationship between the meanings of two paradigms, and will be easiest to create if the paradigms in question are mapped and understood. Paradoxically, two different paradigms attempting to use a common vocabulary may produce more confusion and difficulty than if they had used custom vocabularies, because their paradigmatic differences may be concealed in the use of common terms.

But mechanical transliteration is not the same as translation. Translation involves the expression of the ideas within one paradigm in the language of another, and to translate mechanically is very hard. Defining a true translation mapping may involve different treatment of the same tags in different contexts and requires an understanding of both paradigms. As the need to connect heart to heart between e-businesses increases, there will be more and more demand for comprehensive approaches to creating XML mappings.

To connect from the heart of my e-business to the heart of yours would be impossibly expensive in shared systems without XML, but even with it the system analysis needed to create the translation is a significant task. We should not assume that XML is a panacea, or that the standardization of vocabularies will automatically bring interoperability. XML provides us with a medium to express our understanding of the meaning of data, but we will still have to first discern realities and differences of meanings when we exchange data.

What Matters Is the Meaning

May the efforts of industry groups to define common XML vocabularies flourish and succeed! But should they fail to reach a single, uniform set in every case, if diversity should multiply, no problem. At least the meanings will have been well described, fully annotated and put within reach of analysis, mapping and machine manipulation. What matters is the meaning, not the markup. ☯

AUTHOR BIO

Simon Phipps, IBM's chief Java and XML evangelist, was part of the team that recommended Java to IBM in 1995. He has since spoken internationally on the new world that is engulfing computing, powered by Web and Java technologies. He now has worldwide responsibility for XML evangelism for IBM. With over 20 years' experience in the computer industry, Simon has worked on networking, data communications and operating systems for various companies in many contexts, including development of the earliest commercial collaborative conferencing software with IBM. (Opinions expressed in this article are those of the author and do not necessarily reflect the views of IBM.)

S.PHIPPS@UK.IBM.COM

Ad

IBM

www.ibm.com/developerworks