

# XML JOURNAL

The World's Leading XML Resource

Volume: 1 Issue: 4

XML-JOURNAL.COM

**XML DevCon FALL 2000**

November 12-15, 2000

Announcing...

December 3-5, 2000

**Wireless DevCon**

## FROM THE EDITOR

**Registering XML**  
by Ajit Sagar pg. 5

## XML INDUSTRY INSIDER

**SOAP and XML**  
by Bob Sutor pg. 24

## XML PROS & CONS

**XML – Friend or Foe?**  
by Zvi Schreiber pg. 46

## 2B OR NOT 2B

**XML: True Collaboration**  
by Coco Jaenicke pg. 52

## XML NEWS

by Malcolm Dean pg. 58

## SYS-CON RADIO

**Interview with Bruce Sharpe**  
from SoftQuad Software Inc. pg. 78

## IMHO

**Bridging the Management Gap**  
by John W. Cocola pg. 82

**SYS-CON MEDIA**

# BUILDING XML MIDDLEWARE USING OMNIMARK

For XML-enabled Internet applications, this language makes sense

by Mark Baker  
see page 34

## Feature: XML: The OMG's XML Metadata Interchange Standard

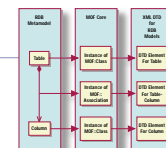
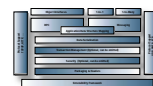
*Using model-centric architecture*

David S. Frankel

6

## XML in Transit: SOAP Part 1

*What is this thing called SOAP? Here's the background*



Simeon Simeonov

12

## Feature: Developing B2B Applications with XML and JMS

*Solving B2B integration problems*

Nirmal Patil & Majeed Ghadialy

18

## Feature: XML Data Interchange

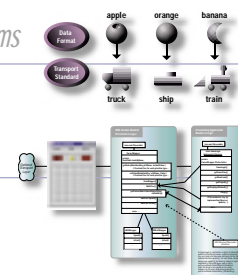
*Choosing a vendor-neutral approach to maximize flexibility*

Mark Wardell

26

## Show Report: XML DevCon 2000

*A report from the conference*

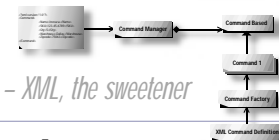


Jim Milbery

32

## Java & XML: Java, XML and the Command Pattern

*Patterns provide the Java recipe – XML, the sweetener*



Israel Hilerio

42

## XML & Business: Eliminating Redundancy Using ID/IDREF

*Mix techniques within the same document*



50

## Java Servlets: New Requirements of Internet Business Architecture

*Develop a single servlet for delivery channels*

Syed Fareed Ahmad

56

# SOFTQUAD SOFTWARE

[www.softquad.com/products/xdmetal/eval](http://www.softquad.com/products/xdmetal/eval)

# PROGRESS

[www.sonicmq.com/ad11.htm](http://www.sonicmq.com/ad11.htm)

# ICON INFORMATION SYSTEMS

[www.xmlspy.com](http://www.xmlspy.com)

**XML****EDITORIAL ADVISORY BOARD**

COCO JAENICKE, SIMON PHIPPS, RICK ROSS, 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: MALCOLM DEAN  
 E-BUSINESS EDITOR: ISRAEL HILERIO  
 JAVA TECHNOLOGY EDITOR: JASON WESTRA

**WRITERS IN THIS ISSUE**

SYED FAREED AHMAD, MARK BAKER, JOHN COCULA, JOHN EVIDEMON,  
 DAVID S. FRANKEL, MAJEED GHADIALY, ISRAEL W. HILERIO, COCO JAENICKE,  
 JIM MILBERRY, JP MORGENTHAU, NIRMAL PATIL, AJIT SAGAR,  
 ZVI SCHREIBER, SIMEON SIMEONOV, BOB SUTOR, MARK WARDELL

**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: \$79/YR (12 ISSUES) CANADA/MEXICO: \$99/YR

ALL OTHER COUNTRIES \$129/YR

(U.S. BANKS OR MONEY ORDERS)

PUBLISHER, PRESIDENT AND CEO: FUAT A. KIRCAALI  
 VICE PRESIDENT, PRODUCTION: JIM MORGAN  
 VICE PRESIDENT, MARKETING: CARMEN GONZALEZ  
 COMPTROLLER: BRUCE MILLER  
 ADVERTISING ACCOUNT MANAGERS: MEGAN RING  
 ROBYN FORMA  
 JDSTORE.COM: AMANDA MOSKOWITZ  
 SYS-CON EVENTS MANAGER: ANTHONY D. SPITZER  
 ADVERTISING ASSISTANT: CHRISTINE RUSSELL  
 ADVERTISING INTERN: ALISON NOVICK  
 GRAPHIC DESIGNERS: ABRAHAM ADDO  
 CATHRYN BURAK  
 GRAPHIC DESIGN INTERNS: AARATHI VENKATARAMAN  
 LOUIS F. CUFFARI  
 ROBERT DIAMOND  
 WEB DESIGNERS: STEPHEN KILMURRAY  
 GINA ALAYYAN  
 WEB SERVICES CONSULTANT: BRUNO Y. DECAUDIN  
 CUSTOMER SERVICE: ELLEN MOSKOWITZ

**EDITORIAL OFFICES**

SYS-CON PUBLICATIONS, INC.  
 135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645  
 TELEPHONE: 201 802-3000 FAX: 201 782-9637  
 SUBSCRIBE@SYS-CON.COM

XML-JOURNAL (ISSN# PENDING)  
 is published monthly (12 times a year) by  
 SYS-CON Publications, Inc., 135 Chestnut Ridge Road,  
 Montvale, NJ 07645

3rd Class Postage rates are paid at  
 Montvale, NJ 07645 and additional mailing offices.

POSTMASTER: Send address changes to:

XML-JOURNAL, SYS-CON Publications, Inc.,  
 135 Chestnut Ridge Road, Montvale, NJ 07645.

**©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**

730 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.

**SYS-CON  
MEDIA**

WRITTEN BY AJIT SAGAR EDITOR-IN-CHIEF ]

from  
the  
editor  
to  
the  
editor

# Registering XML

My wife gave birth to a baby boy in May. When my mother called from India to congratulate us, she told me: "We were thinking of nicknaming him 'Java,' but that sounds feminine. So we decided to call him 'XML' after your magazine." She was kidding, of course. However, the thought of XML becoming a household name that even Mom can use so casually is, to say the least, unsettling. I wonder if I can register his vital statistics with OASIS as an XML schema. More on XML schemas and repositories later.

XML DevCon, held in New York City in late June, was the largest XML event ever – 3600-plus attendees. The show was totally sold out. Kudos to the folks at SYS-CON and Camelot Communications who made it happen. Missed it? Couldn't travel from Silicon Valley to Wall Street? Well, we're holding another event just for you. Please check this issue for details on XML DevCon San Jose, November 12–15. I have a feeling this follow-up conference will be even bigger – and probably sold out too. Please register early if you plan to attend.

**XML  
DevCon  
FALL  
2000**

## XML Repositories and Registries

My colleague said something to me last week that applies so well to the XML industry today: "Last year it was all about private exchanges and portals. Now it's all about consortia." This is so true. He was referring to the B2B industry in general. And as you know, B2B is intimately related to XML. XML provides a standard for defining data formats for transportation across the Web. XML data format types are expressed in the form of XML schemas. An XML schema is a document that describes a set of XML document instances. In that sense it's like an XML document template. The only way that enterprises will agree on common schemas is if there's a shared resource that makes the same schema available to multiple organizations. Such resources should be governed by industry consortia so that multiple organizations can be represented and the acceptance criteria can be as unbiased as possible.

However, shared resources need careful management. As different industry verticals define unique schemas for exchanging XML-based information, the proprietary nature of these schemas will lead to a lack of portability across different e-business environments. There will also be an explosion in the number of redundant schemas that will emerge to express the same type of data. There is a growing need in e-businesses for compatible processes and vocabularies to reduce this redundancy and the consequent complexity. For industries to exchange data using XML across multiple enterprises, standard repositories are



—continued on page 31



AJIT @SYS-CON.COM

**AUTHOR BIO**

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

[ WRITTEN BY DAVID S. FRANKEL ]

Over the past few years the OMG has created an architecture for managing metadata. This has resulted in the issuance of several official metadata standards. The core standard is the Meta Object Facility (MOF), and XML Metadata Interchange (XMI) is an extension of the MOF into the XML space. Thus, before focusing on XMI, it's important to grasp the basic concepts of the MOF.

The MOF proceeds from the fundamental proposition that the best way to describe metadata is to rigorously model its structure using object-oriented technology. The OMG's Unified Modeling Language (UML) is the industry-standard medium for OO modeling. Modelers typically use it to model business, GUI and infrastructure objects. The MOF standard selects a subset of UML that's appropriate for modeling metadata. This subset is called the MOF Core.

Consider a relational database system (RDB). The metadata in an RDB consists of definitions of tables, columns and more. Data models are sets of specific tables and columns. A typical (oversimplified!) data model might define both a Customer and an Account table. The former would have columns such as customer number, name, address and telephone number; the latter would have columns such as number, name and balance. This particular data model isn't the same thing as Customer and Account data. Instead, it describes the structure of the data. Therefore, the table and column definitions that make up the data model are called *metadata* – data about data.

A formal model of metadata is called a *metamodel*. In the previous issue of *XML-Journal* (Vol. 1, issue 3) the **Objects & XML** column contained a simple RDB metamodel that we'll use as our example. The MOF Core contains most of the UML constructs for expressing class models – classes, associations and subtyping. Therefore, UML modeling tools can be used to describe metamodels. Figure 1 uses UML notation to describe the simple RDB metamodel.

If you're a UML aficionado you're probably curious to know which UML constructs used for class modeling aren't included in the MOF Core. They are:

- **AssociationClasses** (associations that are first-class objects)
- **Qualifiers**
- **N-ary associations** (associations among more than two classes)

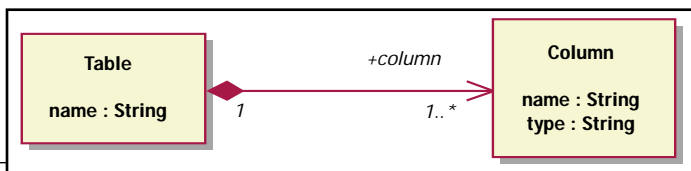


FIGURE 1 Simple RDB metamodel

The reason for not including all the UML class modeling constructs in the MOF Core is that the MOF architects wanted to keep the MOF simple and its footprint small. Metamodels tend to be less complex than models of business, GUI and infrastructure objects, so the fancier UML concepts were omitted.

# XMI: THE OMG'S XML METADATA INTERCHANGE STANDARD



The MOF also defines a set of rules for mapping the elements of the MOF Core to CORBA IDL. This means that given a metamodel, IDL can be generated that specifies the interfaces for CORBA objects that represent models in a repository. The last ***X**ML-J* issue showed the IDL for the RDB metamodel that was generated according to these rules. Say I want to store a data model consisting of a Customer and an Account table. These tables would be represented in a CORBA repository by CORBA objects that expose the Table interface generated from the metamodel by the MOF-IDL mapping. The name, address and other Columns would be represented in a CORBA repository by CORBA objects that expose the Column interface generated from the metamodel. The Column interface generated from the metamodel.

Thus it isn't necessary for a human to hand-code IDL for an MOF-compliant metamodel. The IDL is determined by the metamodel.

The key point is that the MOF Core is independent of CORBA, Java, XML or any other middleware or conveyance technology. Achieving this independence was quite easy since UML, of which the MOF Core is a subset, is technology-neutral. The approach of mapping each of the MOF Core's constructs – classes, associations and so on – to IDL technology can be applied to produce mappings to other technologies so that artifacts in line with these technologies can be generated as well.

### Enter XMI

The MOF was defined before XML became popular. (We tend to forget how recent a phenomenon XML is.) However, the technology-neutral nature of the MOF Core made it relatively easy to produce a mapping from the MOF Core's elements to XML so that, given a metamodel, a Document Type Definition (DTD) could be generated. This DTD can be used to stream models that conform to the metamodel. For example, the DTD generated for our simple RDB metamodel would be used to exchange specific RDB data models.

The generated DTD defines XML elements for each element of the metamodel. For example, the DTD generated for our RDB metamodel defines an element corresponding to Table. That element describes how to convey a representation of a Table definition in an XML stream.

XMI is the OMG standard that defines the rules for generating an XML DTD from a metamodel. The current official version of XMI is 1.1.

### Under the Hood

As described above, the key elements of the MOF Core are classes and associations. We use these elements to describe our simple RDB metamodel. In UML notation each element of the RDB metamodel is an instance of one of the MOF Core's elements (see Figure 2). Thus Table and Column are instances of MOF::Class, while the association between Table and Column is an instance of MOF::Association.

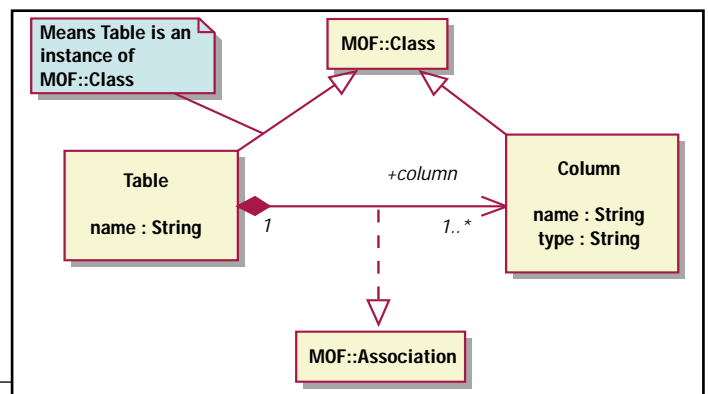


FIGURE 2 Elements of the RDB metamodel are instances of elements of the MOF Core

Figure 3 is a more informal overview of how XMI's MOF-XML mapping is applied to our RDB metamodel. The MOF-XML mapping defines how instances of MOF::Class and MOF::Association are mapped to DTD elements. Of course, the mapping for instances of MOF::Class are different from (but related to) the mappings for instances of MOF::Association.

# Using model-centric architecture

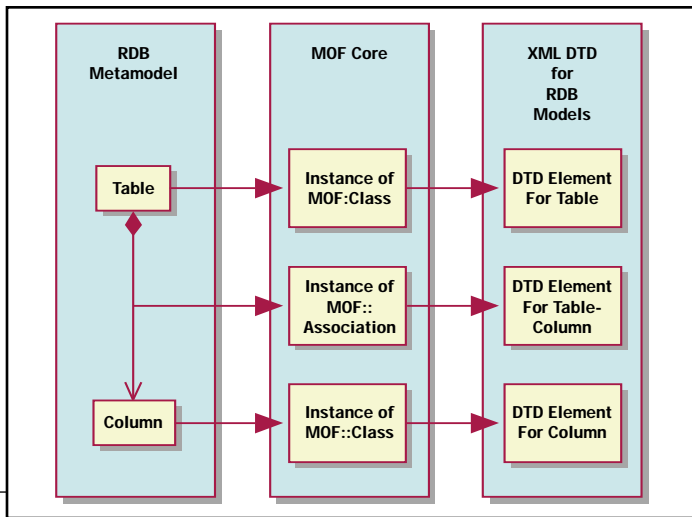


FIGURE 3 Applying the MOF-XML mapping to the simple RDB metamodel

Listing 1 is the main part of the XMI DTD generated for our simple RDB metamodel. This DTD was generated by IBM's alphaWorks XMI Toolkit, available on the alphaWorks Web site ([www.alphaWorks.ibm.com](http://www.alphaWorks.ibm.com)). I've omitted a rather lengthy boilerplate header that makes up the first part of all XMI DTDs and isn't very interesting. Note that the navigable end of the association between Table and Column (the end with the arrowhead in Figure 1) is considered a property of the Table element. The metamodel names the navigable association end *column*, and that name is used in the DTD.

## Rendering OO Models to Non-OO DTDs

The MOF Core, as a subset of UML, is object oriented; XML is not. In particular, XML doesn't allow subtyping (inheritance). Good object models make liberal use of subtyping. The architects of XMI wished to avoid having DTD elements repeat all the properties of all their ancestors since that would make it quite cumbersome to render typical object models to DTDs.

Figure 4 alters our simple RDB metamodel to use subtyping. The name attribute is abstracted out into a supertype named *ModelElement*. (According to UML notation, the name of this class is italicized in the diagram to denote that it is abstract. An abstract class can't have instances that aren't instances of one of its subtypes.)

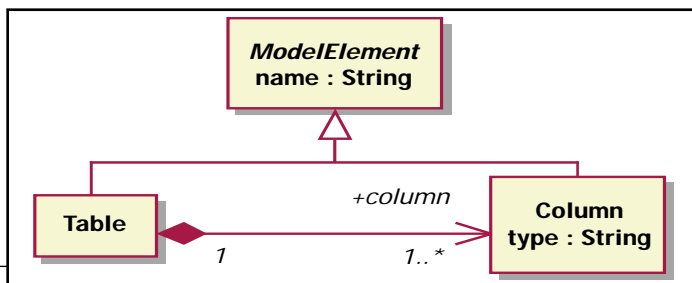


FIGURE 4 Simple RDB metamodel using subtyping

Listing 2 demonstrates how an optional XMI feature avoids repeating a supertype ELEMENT's properties in ELEMENTs representing subtypes. This DTD was generated by a beta version of Unisys's pure Java metadata repository product called *Complex Information Manager*. For each class in the metamodel an ENTITY declaration that encompasses all the attributes of the supertype is generated. Subtypes need to reference only the ENTITY to essentially include all the inherited attributes. In this simple example the ENTITY declaration includes only one attribute because, in the metamodel, *ModelElement* has only one. *Note:* If a class is a subtype of another class (which is typical in a full-scale metamodel), then its ENTITY declaration also encompasses the ENTITY declaration of its supertype.

Since XMI specifies alternate rule sets for generating DTDs, they can be generated with or without these extra ENTITY declarations. The rule set used to generate a DTD doesn't have any impact on what streams can validate against it. The DTD in Listing 1 was generated with a different rule set, so it doesn't have the extra ENTITY declarations.

For various technical reasons separate ENTITY declarations are generated to encompass attributes and associations, respectively. Therefore, a DTD generated from an industrial-strength metamodel using the ENTITY-producing rule set usually contains a substantial sprinkling of ENTITY declarations. When XML-oriented people look at XMI DTDs, the ENTITY declarations tend to be disconcerting. While these DTDs are perfectly legal XML, they look unnatural to the trained XML eye. However, it's important to understand that the XML documents that validate against such DTDs don't look unnatural. The ENTITY declarations are shorthand for the DTD that makes the DTD more compact, if a bit strange looking. There's no way to avoid copying all the subtypes' properties in the XML document itself, so nothing looks out of the ordinary in an XMI-compliant XML document.

## Some Industry-Standard XMI DTDs

Several specific XMI DTDs have been standardized by the OMG. Each was produced by feeding an MOF-compliant metamodel into an XMI DTD generator.

### UML

The UML DTD is the most widely used XMI DTD. Many people think of UML only as a notation, but the official specification has a complete MOF-compliant metamodel. It's MOF-compliant because its elements are defined via the constructs of the MOF Core. This metamodel was fed into an XMI DTD generator to produce the UML DTD used by tools to export and import UML models.

Figure 5 shows the central part of the UML metamodel. It's actually more extensive than this class diagram suggests, but most of the elements not shown here derive in some way from these central elements.

Listing 3 is a fragment of the UML XMI DTD corresponding to the Classifier element of the UML metamodel. Note the property slot for *isAbstract* in the Classifier ELEMENT, which is fully qualified based on the name of the ancestor (GeneralizableElement) in which it was defined in the metamodel. This DTD was generated via a rule set that eschews the ENTITY declarations; that's why the *isAbstract* slot is copied down to the subtype hierarchy and appears directly in the Classifier ELEMENT.

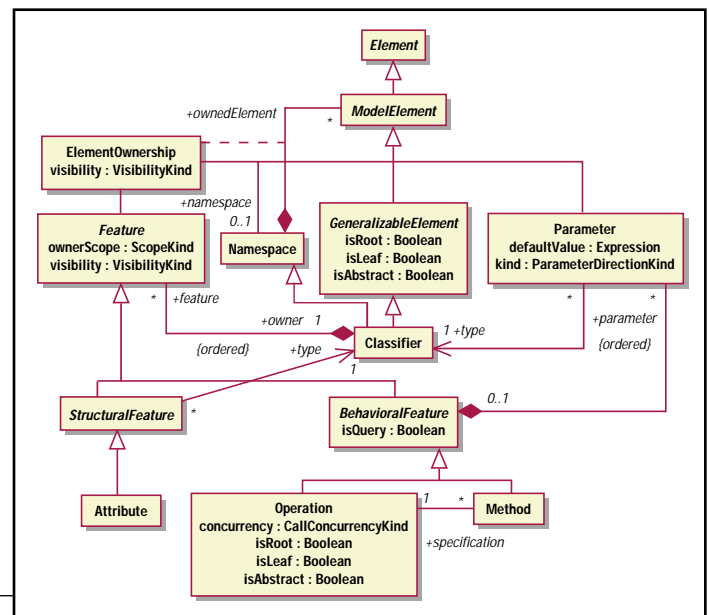


FIGURE 5 The central part of the UML metamodel



# INFOSHARK

[www.infoshark.com](http://www.infoshark.com)

## CORBA COMPONENTS

The CORBA Component Model (CCM) includes two MOF-compliant metamodels. One is a metamodel of CORBA IDL that includes some new IDL constructs defined by the CCM specification. It generates an XMI DTD that's used to exchange CORBA object models that could also be expressed in CORBA IDL.

The other CCM metamodel is for the packaging and deployment descriptors. A CCM component is deployed with a set of these descriptors expressed in XML. The XMI DTD generated from the packaging and deployment metamodel defines the structure of XML documents that contain these descriptors. Figure 6 shows a small portion of the descriptor metamodel.

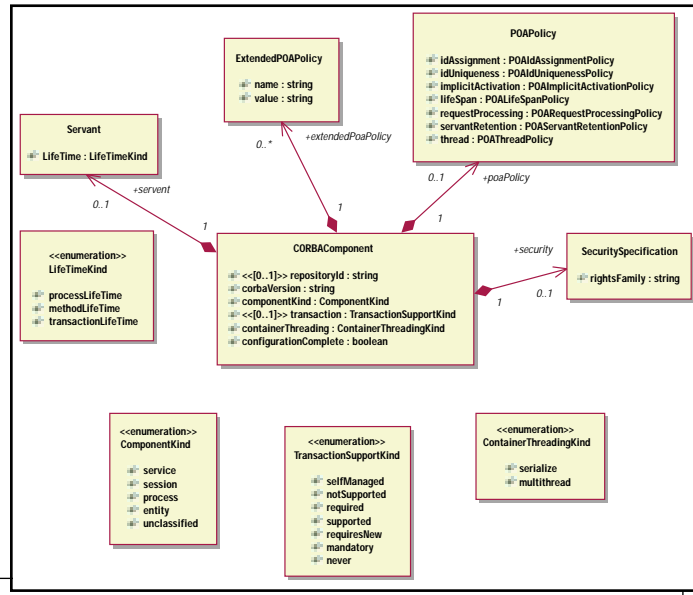


FIGURE 6 Part of the CORBA-component packaging and deployment descriptor metamodel

### LISTING 1

```
<!ELEMENT Table.col (Column)* >
<!ELEMENT Table.name (#PCDATA | XMI.reference)* >
<!ELEMENT Table (Table.name?, XMI.extension*, Table.col*)? >
<!-- ===== RDB: Table ===== -->
<!-- ===== RDB: Column ===== -->
<!-- ===== RDB: Model Element ===== -->
<!-- ===== RDB: Model Element Features ===== -->
<!-- ===== RDB: Model Element Attributes ===== -->
<!-- ===== RDB: Table ===== -->
```

### LISTING 2

```
<!-- ===== RDB: Model Element ===== -->
<!-- ===== RDB: Model Element Features ===== -->
<!-- ===== RDB: Model Element Attributes ===== -->
<!-- ===== RDB: Table ===== -->
```

The ComponentKind is the element in the component descriptor metamodel that supports designating a component as an entity, process, session or service. Figure 6 shows the declaration of ComponentKind as an enumeration and the Component element with an attribute of type ComponentKind. Listing 4 shows fragments of the XML DTD generated from the descriptor metamodel. These fragments correspond to the metamodel's ComponentKind element and tell us precisely how to encode a component to be an entity, process, session or service.

## COMMON WAREHOUSE

The OMG has adopted the Common Warehouse Metamodel as a standard metamodel for relational, record-oriented and hierarchical databases and for other aspects of data warehousing. Unisys, IBM, Oracle, Hyperion, Genesis and other companies responsible for CWM followed the MOF architecture so that XMI DTDs used for exchanging warehouse models could be generated from the metamodel.

Space doesn't permit showing the CWM metamodel, but suffice it to say that it's comprehensive. Listing 5 is a fragment of the XMI DTD generated from the metamodel for relational databases.

## What Does "XMI Compliant" Mean?

When a vendor claims XMI compliance, you must dig a bit to find out exactly what this means. One level of compliance is support for some particular XMI DTD. An example is Rational Rose, which supports the UML XMI DTD. In addition, as the CWM is incorporated into products, support for any of the CWM XMI DTDs would be another example of this kind of compliance.

On the other hand, a product might support generating a DTD from a metamodel – a different kind of compliance. Furthermore, one vendor might support the rule set that generates the extra ENTITY declarations, another might not. Yet another might give you the choice of using either rule set.

## Using XMI with Instance Data

Although XMI was originally designed for metadata exchange, we can also use it to exchange instance data. Consider the UML model in Figure

```
<!-- ===== RDB: Table ===== -->
<!-- ===== RDB: Column ===== -->
<!-- ===== RDB: Model Element ===== -->
<!-- ===== RDB: Model Element Features ===== -->
<!-- ===== RDB: Model Element Attributes ===== -->
<!-- ===== RDB: Table ===== -->
```


### LISTING 3

```
<!-- ===== RDB: Table ===== -->
<!-- ===== RDB: Column ===== -->
<!-- ===== RDB: Model Element ===== -->
<!-- ===== RDB: Model Element Features ===== -->
<!-- ===== RDB: Model Element Attributes ===== -->
<!-- ===== RDB: Table ===== -->
```

### LISTING 4

```
<!-- ===== RDB: Table ===== -->
<!-- ===== RDB: Column ===== -->
<!-- ===== RDB: Model Element ===== -->
<!-- ===== RDB: Model Element Features ===== -->
<!-- ===== RDB: Model Element Attributes ===== -->
<!-- ===== RDB: Table ===== -->
```

## Conclusion

The specification of a platform-independent model (or metamodel) as the source for generating a DTD is part of an overall architectural approach that also uses a platform-independent model as the source for the production of artifacts for other technology environments. The other environments include CORBA, EJB and LDAP. This approach has been dubbed *model-centric architecture*. The model distills the essential information semantics from platform concerns so they can be readily reused in different platform environments. Furthermore, model-centric architecture allows many platform-specific artifacts, including XML DTDs, to be automatically generated by generic tools. This increases the degree of automation that can be applied to the production of DTDs and other artifacts, thus enhancing the scalability of these technologies. 

## References

- *alphaWorks XMI Toolkit*: [www.alphaworks.ibm.com/tech/xmitoolkit](http://www.alphaworks.ibm.com/tech/xmitoolkit)
- *Common Warehouse Metamodel (CWM) Specification, February 11, 2000*: <http://cgi.omg.org/cgi-bin/doc?ad/00-01-01>  
<http://cgi.omg.org/cgi-bin/doc?ad/00-01-02>
- *Meta Object Facility (MOF) Specification, Version 1.3, September 27, 1999*: <http://cgi.omg.org/cgi-bin/doc?ad/99-09-05>
- *Unified Modeling Language (UML) Specification, Version 1.3, June, 1999*: <http://cgi.omg.org/cgi-bin/doc?ad/99-06-08>
- *Unisys Universal Repository*: [www.urep.com/marketplace/urep](http://www.urep.com/marketplace/urep)
- *XML Metadata Interchange (XMI) Specification, Version 1.1, October 25, 1999*: <http://cgi.omg.org/cgi-bin/doc?ad/9910-02>  
<http://cgi.omg.org/cgi-bin/doc?ad/99-10-03>

## AUTHOR BIO

David S. Frankel is chief scientist at Genesis Development Corporation ([www.gendev.com](http://www.gendev.com)), West Chester, Pennsylvania, an IONA Technologies company. He's also a member of the OMG Architecture Board and cochair of OMG's Business Object Initiative Working Group.

DAVIDFRANKEL@IONA.COM

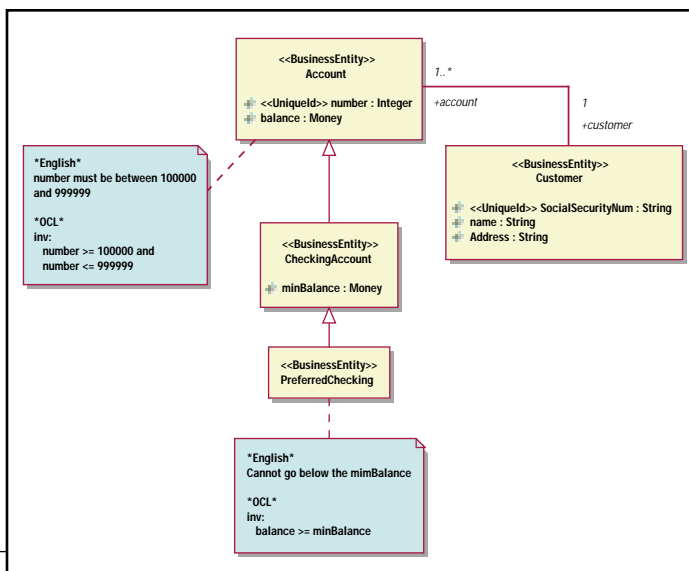


FIGURE 7 Simple UML model of a bank – information viewpoint

7. This isn't a metamodel. It's a model expressed via the UML metamodel. Instances of these classes constitute real user objects, not models. The model is platform-independent; that is, it's not specific to CORBA, EJB, XML or any other platform or conveyance technology.

Since the MOF Core is very close to UML, we can apply the XMI algorithms to a UML model, such as our bank model, to generate a DTD that tells us how to encode instances of the banking entities in an XMI stream. If we submit our UML bank model to an XMI DTD generator, we get a DTD that tells us how to encode the state of our instances, such as the state of specific Accounts and Customers. Thus the DTD for exchanging instance data can be derived from the formal UML model. Listing 6 is a fragment of the DTD.

```
-->
...
<! ELEMENT Component.CORBAComponent.componentKind EMPTY>
<! ATTLIST Component.CORBAComponent.componentKind
%Component.ComponentKind;
```

### LISTING 5

```
<!-- CLASS: CWMRDB:Table -->
<! ELEMENT CWMRDB:Table.isSystem EMPTY>
<! ATTLIST CWMRDB:Table.isSystem
xmi.value ( true | false ) #REQUIRED>
<! ELEMENT CWMRDB:Table.optionScopeColumn (CWMRDB:Column) *>
<! ELEMENT CWMRDB:Table.type (CWMRDB:SQLStructuredType)?>
<! ELEMENT CWMRDB:Table.usingTrigger (CWMRDB:Trigger) *>
...
```

### LISTING 6

```
<!-- Customer -->
<! ELEMENT Bank.Customer.SocialSecurityNum (#PCDATA |
XMI.reference) *>
<! ELEMENT Bank.Customer.name (#PCDATA | XMI.reference) *>
<! ELEMENT Bank.Customer.Address (#PCDATA | XMI.reference) *>
<! ELEMENT Bank.Customer.account (Bank.Account |
Bank.SavingsAccount |
Bank.CheckingAccount |
Bank.PreferredChecking |
Bank.RegularChecking) *>
<! ELEMENT Bank.Customer
(Bank.Customer.SocialSecurityNum?,
Bank.Customer.name?,
```

```
Bank.Customer.Address?,
XMI.extension*,
Bank.Customer.account*)?
>
```

```
<! ATTLIST Bank.Customer
%XMI.element.att;
%XMI.link.att;
>
<!-- Account -->
<! ELEMENT Bank.Account.number (#PCDATA | XMI.reference) *>
<! ELEMENT Bank.Account.balance (#PCDATA | XMI.reference) *>
<! ELEMENT Bank.Account.customer (Bank.Customer)?>
<! ELEMENT Bank.Account
(Bank.Account.number?,
Bank.Account.balance?,
XMI.extension*,
Bank.Account.customer?)?
>
<! ATTLIST Bank.Account
%XMI.element.att;
%XMI.link.att;
>
```





*What is this thing called SOAP? Here's the background*

# SOAP

Part 1

I wanted to kick off this new column on XML protocols with an introduction to the hot newcomer in this arena – Simple Object Access Protocol (SOAP). Trouble is, there are too many ways to go about the topic. The first version I wrote was a technical introduction to SOAP laced with references to some of the important ongoing debates on XML protocols and XML distributed computing. When I finished I found that the two threads – detailed technical information and higher-level issues – interfered with each other. So I decided to focus on the technology in the next issue and devote this space to a discussion of the driving forces behind SOAP.

## The Making of SOAP

The following story is pieced together from what I've learned from information volunteered by employees and alumni of Microsoft, DevelopMentor and UserLand (the initial creators of SOAP) as well as Allaire, DataChannel, IBM, W3C and webMethods (other active players in the XML distributed computing space). I welcome additions and/or corrections.

Microsoft started thinking about XML distributed computing in 1997; SOAP was coined in early 1998. The goal was to enable applications to communicate via Remote Procedure Calls (RPCs) on top of HTTP. DevelopMentor (a long-standing Microsoft ally) and UserLand (a company that saw the Web as the ultimate publishing platform) joined the discussions. Things moved forward, but as the group tried to involve wider circles, at Microsoft politics stepped in and the process stalled. The DCOM camp at the company disliked the idea of SOAP and believed that Microsoft should use its dominant position in the market to push the DCOM wire protocol via some form of HTTP tunneling instead of pursuing XML. Some XML-focused folks at Microsoft believed the SOAP idea was good but had come too early. Perhaps they were looking for some of the advanced facilities that could be provided by XML schema and name-

spaces (see my article, "The Evolution of XML Protocols," in *XML-J*, Vol. 1, issue 3). Frustrated by the deadlock, UserLand went public with a cut of the spec published as XML-RPC in the summer of 1998.

In 1999, as Microsoft was working on XML Data and adding support for namespaces in its technologies, the idea of SOAP gained additional momentum. It was still an XML-based RPC mechanism, however. That's why it met with resistance from the BizTalk ([www.biztalk.org](http://www.biztalk.org)) team. The BizTalk model was based more on point-to-point messaging than RPCs. It took people a few months to resolve their differences. In December 1999 SOAP 1.0 was submitted to the Internet Engineering Task Force (IETF). DevelopMentor started an open effort to create a Java SOAP implementation.

In typical fashion the Microsoft marketing/PR machine proudly waved the SOAP banner at every opportunity. SOAP was heralded as the effort that would immediately break down barriers to interoperability and shepherd us into a new era. Given Microsoft's positioning, it wasn't surprising that companies such as IBM and Sun responded very defensively and denounced SOAP as all hype and no substance.

Luckily, common sense prevailed quickly. Microsoft changed its positioning more along the lines of "SOAP is a first step toward the establishment of stan-

dards for a ubiquitous XML distributed computing infrastructure." A few months ago, in the beginning of May, SOAP 1.1 was submitted as a Note to the W3C with IBM as a coauthor. This was an unexpected and refreshing change. In addition, the SOAP 1.1 spec was much more extensible, eliminating concerns that backing SOAP implied backing some Microsoft proprietary technology. This, and the fact that IBM immediately released an open Java SOAP implementation that was subsequently donated to the Apache XML Project ([xml.apache.org](http://xml.apache.org)) for open source development, convinced even the greatest skeptics that SOAP is something to pay attention to. Sun reversed its stance and voiced support for SOAP in the beginning of June. Not long after, Microsoft released a base-level SOAP implementation and announced that the new version of BizTalk is going to be based entirely on SOAP.

That's the status as of this writing. To keep your finger on the pulse of SOAP, you should join the XML distributed applications mailing list at the W3C ([xml-dist-app@w3.org](mailto:xml-dist-app@w3.org)), the SOAP discussion mailing list at DevelopMentor ([soap@discuss.develop.com](mailto:soap@discuss.develop.com)) and the SOAP mailing lists at the Apache XML Project ([soap-user@xml.apache.org](mailto:soap-user@xml.apache.org) and [soap-dev@xml.apache.org](mailto:soap-dev@xml.apache.org)).

## What Should SOAP Do?

SOAP claims to be a specification for a ubiquitous XML distributed computing infrastructure. It's a nice buzzword-compliant phrase, but what does it mean? Let's parse it bit by bit to find out what SOAP should do.

XML surely means that SOAP is based on XML and related standards:

# SILVERSTREAM

[www.silverstream.com](http://www.silverstream.com)

Facet	Possibilities	Extensible	Pluggable
Protocols	TCP/IP, EDP, HTTP, SMTP, FTP	✓	
Security	None Username/password X.509 certificates	✓	✓
Transactions	None Two-phase commit	✓	✓
Data marshaling	At least one default	✓	✓
Communication pattern	Point-to-point Publish/subscribe RPC	✓	

TABLE 1 Distributed computing scenarios that SOAP should be able to support

namespaces, schema, and so on. So far so good.

*Distributed computing* implies that SOAP can be used to enable the interoperability of remote applications (in a very broad sense of the phrase). It's a fuzzy term and it means different things to different people and in different situations. Here are some "facets" one can use to think about a particular distributed computing scenario: the protocol stack is used for communication, connection management, security, transaction support, marshaling and unmarshaling of data, protocol evolution and version management, error handling and audit trails. The actual requirements for different facets will surely vary between scenarios. For example, a stock ticker service that continuously distributes stock prices to a number of subscribers will have needs different from those of an e-commerce payment processing service. The stock ticker service will probably need no support for transactions and only minimal – if any – security or audit trails (it distributes publicly available data). The e-commerce payment processing service will require Cerberian security, heavy-duty transaction support and full audit trails.

*Infrastructure* implies that SOAP is aimed at low-level distributed systems developers, not developers of application/business logic or business users. Infrastructure products such as application servers will become "SOAP enabled." SOAP will work behind the scenes making sure your applications can interoperate without your having to worry too much about it.

*Ubiquitous* means omnipresent, universal. On first look it seems to be a

meaningless term, thrown into the phrase to make it sound grander. It turns out, however, this is probably the most important part. The ubiquity goal of SOAP is both a blessing and a curse. It's a blessing because if there are SOAP-enabled systems everywhere on the Internet, it should be easier to do distributed computing. After all, that's what SOAP is all about. However, the ubiquity of SOAP is also a curse because one technology specification should be able to support many different types of distributed computing scenarios, from the stock ticker service to the e-commerce payment processing service. To meet this goal SOAP needs to be a highly abstract and flexible technology. However, the more abstract SOAP gets, the less support it will provide for specific distributed computing scenarios. This is the eternal tug of war between generality and specificity.

Let's look back at some of the facets of distributed computing and make a best guess about the bare minimum in variations of distributed computing scenarios SOAP should be a base for. Table 1 could be considered a start in this

direction. *Extensible* means that someone can extend how a facet of a new scenario can be handled without having to modify the core specification. *Pluggable* means that the specification allows for introducing a completely new mechanism for handling a certain facet.

Wow! What are the chances that a single specification can address the details of all these scenarios? Pretty darn close to zero if you ask me. That's why I can't stress enough that SOAP (at least in its current form) is just a start, a base specification that addresses problems common to all distributed computing scenarios. Saying that you're going to use SOAP for XML distributed computing is like saying you're going to use XML for structured data representation or ASCII for writing text. It's a fairly meaningless statement until you specify a whole lot more: protocols, security, transactions, data marshaling, communication patterns, and so on. Be realistic; don't let the SOAP hype take over!

## What Is SOAP, Really?

Despite the hype, SOAP is still of great importance because it's the industry's best effort to date to standardize on the infrastructure technology for XML distributed computing. There is general consensus in the industry that SOAP should be focused on the common aspects of all distributed computing scenarios and therefore needs to provide:

- *An extensibility mechanism* so that evolution isn't hindered and there's no lock-in. XML, schema and namespaces really shine here.
- *Packaging of information* in a clearly identifiable SOAP "request" or "message." This is done via a SOAP "envelope" that encloses all other information.
- *Identification of the "body" of the request/message* where potentially arbitrary XML could be used.
- *Communication of additional information*



The future of SOAP and specifications related to it is uncertain because of the usual politics and conflicting goals of the major players and standards organizations





# CAPE CLEAR

[WWW.CAPECLEAR.COM](http://WWW.CAPECLEAR.COM)

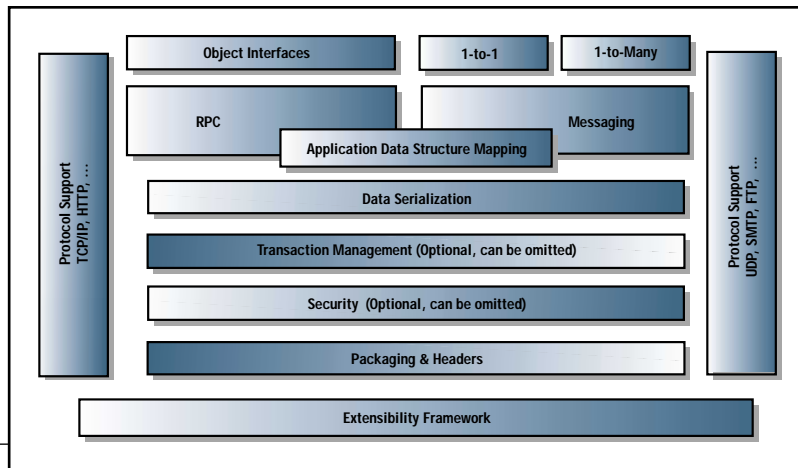


FIGURE 1 One possible layering of SOAP and related specifications

tion that sits outside the body (via a notion of "headers"). Headers can be used to build more complex protocols on top of SOAP.

- **Basic error handling.** This is done via the notion of a SOAP "fault."
- **Data serialization.** SOAP specifies a default mechanism for data serialization, but others can be added.

Unfortunately, there's considerably less consensus on how higher-level issues should be addressed. Nearly everyone agrees, however, that to tackle the broad spectrum of interesting problems we're faced with we need to work in parallel on a set of layered specifications for XML distributed computing.

Among other things, the XML Protocols Shakedown panel at WWW9 in May focused heavily on the scope of SOAP and on what other specifications we need to put in place to make the technology truly usable. Dan Connolly of the W3C chaired the panel. Present among the panelists and audience were most of the XML distributed computing gurus of our time. Many in the group tended to agree that the important specific problem areas that face us are:

- *Basic RPC, the common model for exposing application "services"*
- *Basic point-to-point and publish/subscribe messaging, the dominant mechanism for loosely coupled application integration*
- *Programming language data structure mapping, à la WDDX, to enable logical data interoperability*
- *Robust security, without which most businesses won't adopt XML distributed computing*
- *Full transaction management, required by mission-critical services such as e-commerce-related technologies*

Please, heed this warning: If we don't try to establish standards in these areas, the benefits of using SOAP will be severely limited. While the extensibility of SOAP is a great thing, it also means that 10 companies can come up with 10 ways to do messaging/security/whatever with it and interoperability will be lost. Don't get me wrong, I'm not a standards fascist. However, I do believe that if the whole industry is faced with common problems, we'll all benefit by sharing common yet extensible standards. Commonality enables interoperability. And extensibility is key because it eliminates lock-in.

When faced with the trade-off between generality and specificity, complexity and simplicity, we've generally adopted a combination of layering and parallel development. Layered horizontal specifications can be used to address increasingly broad problem sets. Parallel development of vertical specifications can quickly deliver solutions for separate problem domains. This is likely to be how SOAP and related technologies will evolve. Of the problem areas above, the first two qualify as vertical, the last three as horizontal. Figure 1 illustrates one way they can be organized.

## The Road Ahead

The SOAP development community is working hard. There have been some great discussions on security on the DevelopMentor SOAP mailing list. Allaire, IBM and UserLand are thinking about application data type mapping. Microsoft is toying with the idea of using the Transaction Internet Protocol (TIP, RFC2371) it developed together with Tandem as a means of transaction management. The base SOAP specification includes ideas on how RPCs should be handled, and IBM, Microsoft and

third parties have developed competing specifications for describing object interfaces and Web services that can be exposed via SOAP. Despite this excitement, the future of SOAP and specifications related to it is uncertain because of the usual politics and conflicting goals of the major players and standards organizations.

The major vendors backing SOAP aren't all on the same page. Microsoft stands firmly behind the philosophy of releasing some baseline technology quickly and letting developers run with it. Companies suspicious of Microsoft interpret this as a move to standardize on the base-level SOAP spec and then use this to put a "standards approved" seal on any proprietary WinDNA extensions to SOAP. IBM and Sun haven't communicated a clear strategy except that they're interested in high-end enterprise-quality technology and are likely to push SOAP that way. The smaller players vary in positioning from publishing companies that don't care even about security to e-business platform vendors that want the technology to scale nicely from simple to complex scenarios.

The standards organizations have plenty to think about, too. SOAP was initially submitted as an IETF draft and then as a W3C Note. This move created some discontent. Although the IETF and the W3C have worked together on some projects, in general they're quite protective of their turf. Furthermore, while the base-level SOAP spec fits the scope of W3C quite well, higher-level specs (transactions, messaging, etc.) seem to be clearly outside it. What organization is going to spearhead standardization in these areas? OASIS is one possible choice. However, there's a lot of overlap between OASIS work and Microsoft's BizTalk framework. There's a good chance that Microsoft is going to be upset if OASIS gets involved in a big way.

I hope as an industry we can figure a way out of this mess quickly, because as you've seen, SOAP by itself just doesn't do that much. We need to put a lot more work into it before it becomes usable by the mass market. To see just how little SOAP does (but how well it does it!), in the next **XML in Transit** column we'll look at the technical aspects of the latest SOAP spec. The material from this issue's column will help you put things in perspective and evaluate the design trade-offs. ☛

## AUTHOR BIO

Simeon Simeonov is chief architect at Allaire Corporation.

SIMEON@ALLAIRE.COM

# AIR2WEB

[WWW.AIR2WEB.COM](http://WWW.AIR2WEB.COM)

[ WRITTEN BY NIRMAL PATIL &amp; MAJEED GHADIALY ]

This article explains the use of XML and Java messaging technology in intra-enterprise or inter-enterprise (B2B) applications.

XML technology allows for a common representation of a wide variety of data and is quickly becoming a leading standard for data representation on the Web. It's difficult to imagine any new Web application effort, or data representation effort for that matter, proceeding without involving XML technology.

Java Message Service (JMS) is a specification from Sun that enables Java developers to write enterprise applications that will be portable across various message-oriented middleware (MOM) products. When used together, these two technologies provide a compelling option to develop business-to-business applications. This article is intended for enterprise-solution architects seeking answers to the questions in the B2B integration arena.

### Problem Definition

Let's talk first about B2B. Suppose, in general terms, Enterprise A and Enterprise B would like to collaborate within a certain context. Enterprise A would like to share some data/process information with Enterprise B as this leads to competitive benefits for both of them. This requires a process workflow to span both enterprises. Data needs to be manipulated and transformed not only across the network, but also across several applications. What makes it difficult? Well, the following to start with...

- **Data representation differences:** For example, the way Enterprise A defines a purchase order will be different from the way Enterprise B defines it.
- **Application differences:** Both enterprises may use different applications to run their business. For example, Enterprise A may have standardized internally on Oracle ERP, while Enterprise B may be using SAP.
- **Network protocol differences:** The networks used by the enterprise can be based on different protocols.
- **Platform differences:** Both enterprises may have standardized on different platforms like Sun Solaris, NT or IBM AIX.
- **Security and firewall issues:** Both enterprises may be using firewalls to protect enterprise data.

Most of these difficulties can be categorized as *data-formatting* or *data-transport* problems (see Figure 1). XML solves the former while MOMs solve the latter.

### XML/JMS

XML is different from HTML. HTML has a well-defined set of tags that describe a fixed number of elements. XML, however, is a meta-markup language in which you make the tags you need when you need them. In HTML you wait for the next release, hoping it'll contain the tag you want.

The tags you define can be documented in a Document Type Definition (DTD), the vocabulary and syntax for the language you defined using XML. XML describes structure and semantics, not formatting. What does all this mean to a solution architect challenged by the problems in B2B integration? This article doesn't go into the programming details of XML. However, for the current discussion it's important to note its key benefits:

# DEVELOPING B2B APPLICATIONS USING XML AND JMS

# Solving B2B integration problems

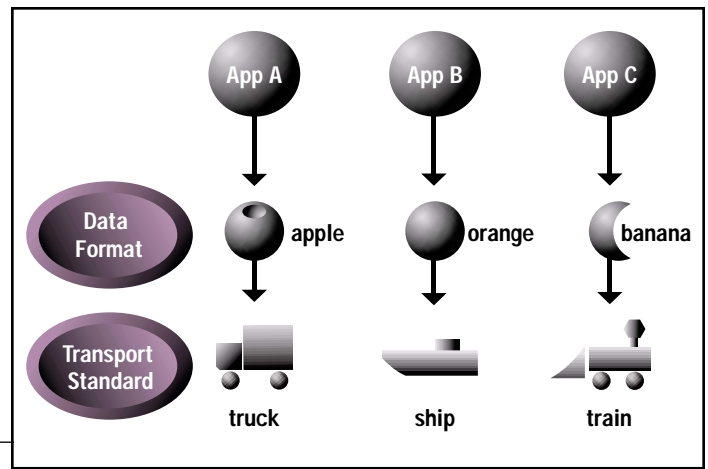


FIGURE 1 How do we integrate?

1. ***XML is a self-describing data format.*** Applications exchanging XML information can infer the content of a data file without having any prior information by looking for the tags in the document.
2. ***XML allows the design of business-specific markup languages.*** Different industry segments can define their own markup languages as a common vocabulary. Examples are Chemical Markup Language (CML), which describes molecular sciences, and Wireless Markup Language (WML), which describes Web content for wireless devices.
3. ***The structure of an XML document is similar to that of a business object with various attributes.*** This allows for the natural conversion of application-specific objects to XML documents and vice versa.

Thus XML is capable of solving data-formatting problems. Let's see how JMS, a specification from Sun, addresses data-connectivity problems.

To connect disparate enterprise applications, it's important to use an architectural approach that facilitates addition, removal and modification of applications without affecting the existing ones. The need for B2B connectivity places strenuous demands on the communication and transaction infrastructure. By adopting a loosely coupled approach, nagging issues such as system maintainability and robustness can be handled effectively. This decoupling goes a long way in building flexible, reliable and highly scalable infrastructures.

MOMs are based on the concept of loosely coupling applications by allowing applications to send guaranteed or reliable messages to one another.

The two major paradigms in messaging are publish/subscribe and point-to-point.

## POINT-TO-POINT MESSAGING

Software applications have unique addresses or associated queue names. The sender application needs to know the identity of the receiver application(s) before it sends messages. The messages may be stored and forwarded by the MOM. This paradigm works fine with a relatively static set of communicating entities.

## PUBLISH/SUBSCRIBE MESSAGING

A set of applications register their interest in different subjects or topics with the MOM. Publishing applications don't have any knowledge of the subscribers; they simply publish messages based on subjects. The MOM forwards a message to the subscriber when it pertains to a subject that is subscribed to. The publishers and subscribers may come and go without affecting any peer applications. This paradigm works fine with a dynamic set of communicating entities. It promotes a high decoupling of applications and scalability of the overall system.

Various vendors have implementations of message-based middlewares. IBM's MQSeries is an example of a MOM and Microsoft has a product called Microsoft Message Queue (MSMQ). Both are based on the point-to-point paradigm. IBM's MQSeries, however, also supports a pub-



lish/subscribe paradigm in version 5.1. Tibco is another vendor of a MOM called TIB/Rendezvous, based on the publish/subscribe paradigm.

JMS is an elegant way to write messaging-based client programs and deploy them on top of any JMS-compliant MOM. JMS makes your application code or business logic code portable across MOMs. This assumes greater importance in the B2B e-commerce scenario as the collaborating enterprises may work with different MOMs. For any enterprise, deploying and maintaining a MOM that constantly carries mission-critical data is a significant investment. This can't be changed overnight because your partner is using another MOM product. Using JMS, the same client program will work fine with both MOM products. Now the business application logic programmer doesn't have to learn the programming APIs for each specific MOM. The end result? Savings both of time and the cost of developing business applications, and quicker time-to-market.

We saw the integration problem in Figure 1. Figure 2 shows the results of introducing XML and JMS.

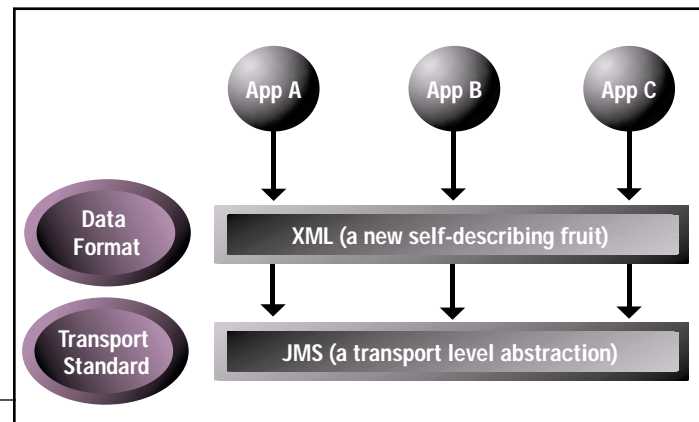


FIGURE 2 The XML/JMS duo

## Application of XML/JMS to B2B

In the health-care industry it's normal to find a highly diverse range of IT infrastructure. Various hospitals have proprietary platforms, with different enterprise resource planning (ERP) applications communicating over various proprietary protocols. They need to submit health insurance claim forms to insurance companies, which have their own set of proprietary data formats and communication protocols. Within each of these insurance companies and hospitals are several software applications that need to interact. And there's a constant need to add new applications and modify the existing ones.

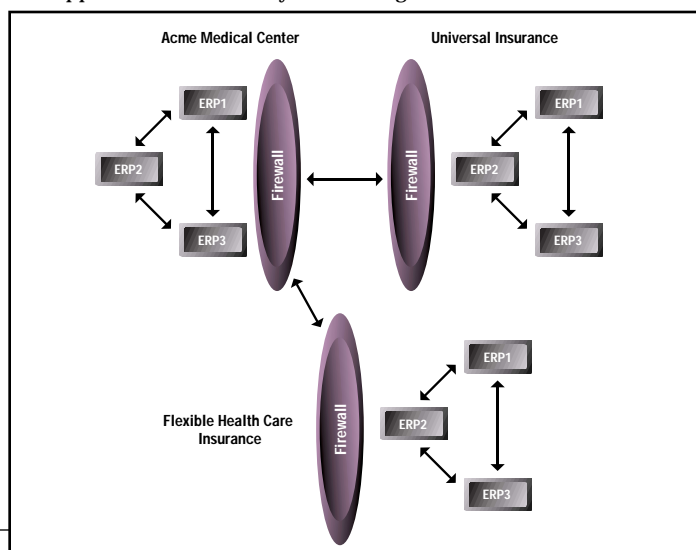


FIGURE 3 Three organizations seeking B2B integration

For this example we'll consider Acme Medical Center: they have three ERP applications and two medical insurance companies, Universal Insurance and Flexible Health Care Insurance, each with three applications. Acme wants to collaborate with both insurance companies and possibly others in the future. These companies need medical claim forms in a format different from what the claim entry system at Acme Medical Center produces. The IT staff at Acme realizes that a future insurance company may need a totally different format.

Another problem with these three organizations is that the ERP applications need to exchange data with each other and all of them need data in a different format (see Figure 3). Although it looks strange, it's typical of an enterprise scenario.

Most enterprise applications were designed with the functional aspects of the system in mind rather than networking needs with other applications.

It's likely that there may be more ERP applications within each of these organizations that need intra-enterprise integration. At the same time, more insurance companies may come on board, also resulting in a need for inter-enterprise integration. It doesn't make sense to do a point-to-point data format conversion and integration for all the cases, since that will result in a large number of custom integrations that need ongoing maintenance and updating. In addition, every time a new insurance company comes on board, a whole new integration will need to be done with all ERP applications inside Acme Medical Center (see Figure 4).

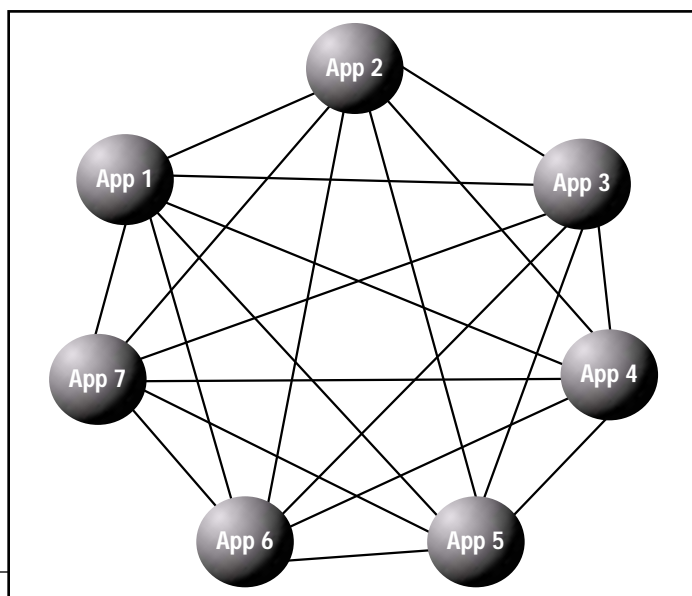


FIGURE 4 Point-to-point integration between several applications

There's a point-to-point integration between every two applications. Think about adding one more application. It isn't going to work. Introduce XML and JMS into the picture and life becomes simpler. As shown in Figure 5, there's only one integration from each application to the central hub. This is achieved by abstracting the data format using XML and

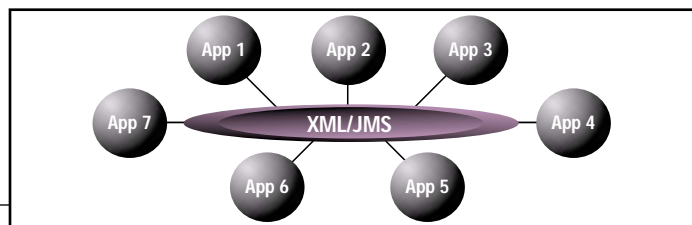


FIGURE 5 Integration using XML and JMS



# THE SHORT LIST

[www.shortlist.com](http://www.shortlist.com)

connecting the applications using JMS. Every application should read and write data in XML format and send and receive it using the JMS API.

## Solution Architecture

The IT department at Acme receives the mandate to come up with a solution architecture that allows seamless integration of various in-house ERP systems, an easy way to add more applications. The architecture should also support easy integration with the existing insurance companies as well as future ones. The IT department at Acme talks to each insurance company's IT department; they weren't surprised to see that both companies were facing similar problems. After weeks of architectural discussions and evaluating various available products, all three IT departments agreed that it would be best to follow open standards (such as XML), a cross-platform language (such as Java) and messaging standards (such as JMS).

## Intra-Enterprise Integration

### MESSAGING PARADIGM

To integrate their internal ERP applications, they go with a publish/subscribe messaging paradigm since it allows them to seamlessly add more applications without doing numerous point-to-point integration.

### DATA FORMAT

They decide that the data format between applications should be a self-describing, extensible format that allows them to accommodate any future applications without modifying existing ones. Hence they opt for XML.

### INTEGRATING XML WITH THE ERP APPLICATIONS

The next problem is integrating XML with each of the ERP applications. They contact the ERP vendors and find that several of them already support XML as a data transfer format. For the remainder of the applications they realize it's not feasible to rewrite them to use the new technologies from the ground up. So they agree to write adapters for each application that will convert XML into the format the ERP applications expect.

## Inter-Enterprise Integration

There are more issues for integration between the organizations.

### SECURITY

Data should be secure over the Internet. This has three aspects:

- **Encryption:** This refers to ciphering of data while on the wire.
- **Authentication:** Only valid applications and users should be allowed to send and receive messages.
- **Data integrity:** Data shouldn't be tampered with while in transit.

### FIREWALL ISSUES

They need to do messaging across each organization's corporate firewall. They could open a port in the firewall; however, for security reasons not every partner likes to do that. The other option is to do what's called HTTP tunneling – sending the data as HTTP traffic through well-known port number 80 for HTTP and then, once inside the firewall, converting this data into messages.

### SELECTION OF MOM

They look at various MOM systems available in the market and decide to go with IBM's MQSeries version 5.1 that supports both point-to-point and publish/subscribe messaging paradigms. It also has an add-on called IBM Internet Gateway that allows messages to go through firewalls using HTTP tunneling. With MQSeries it's also possible to send encrypted messages over the Internet.

### MESSAGING PARADIGM

Although the messaging paradigm within the corporate firewall is publish/subscribe, a point-to-point approach is taken outside simply because no organization wants its messages to be available to others.

For this they decide to develop a simple application that will receive the messages from the partners in a point-to-point fashion and publish it on the message bus. On the outgoing side it'll subscribe to certain subjects that define outgoing messages and send them to one of the partner organizations as point-to-point messages. This is called partner interface application.

The overall solution architecture is shown in Figure 6.

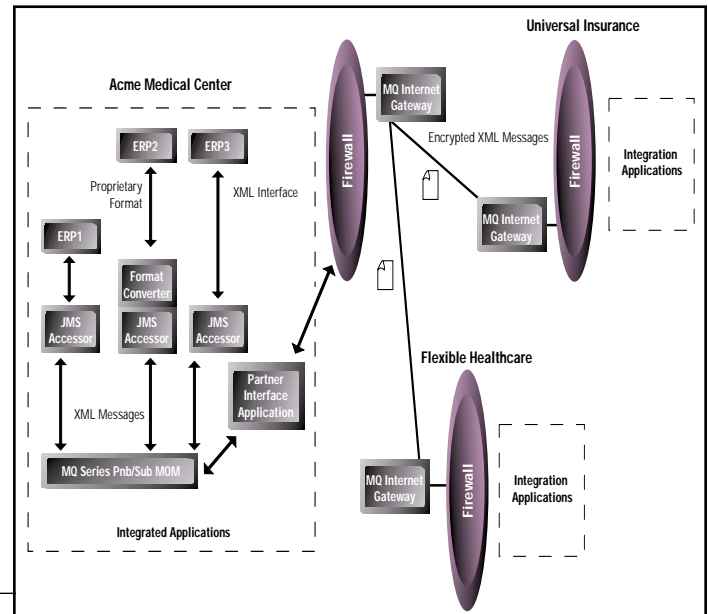


FIGURE 6 B2B story complete with JMS and XML

## Conclusion

XML is a standard that provides portability with respect to various data formats; JMS is a standard that provides transport-level portability. Logically, when these two technologies are applied together, we get solutions to many of the B2B integration problems.

Considering how rapidly these technologies are advancing, don't be surprised to see JMS come up with one more message format, XMLMessage, that allows direct access to and manipulation of an XML document. When that happens, enterprise architects will find this duo even more valuable. ☺

## Further Reading

1. XML Language Specification: [www.w3.org/XML](http://www.w3.org/XML)
2. XML Catalog for Vertical Industry Segments: [www.xml.org/xml-org\\_registry/index.shtml](http://www.xml.org/xml-org_registry/index.shtml)
3. JMS Specification: [www.javasoft.com/products/jms](http://www.javasoft.com/products/jms)
4. IBM MQ Series: [www.software.ibm.com/mqseries](http://www.software.ibm.com/mqseries)
5. Tibco Rendezvous: [www.rv.tibco.com](http://www.rv.tibco.com)
6. Vendors and MOMs supporting JMS: [www.javasoft.com/products/jms/vendors.html](http://www.javasoft.com/products/jms/vendors.html)

## AUTHOR BIOS

Nirmal Patil is a software developer working with i2 Technologies. A Sun-certified Java architect with four years of Java experience, he focuses on designing and developing distributed object systems. Nirmal holds an MS in computer science.

Majeed Ghadialy, a Sun-certified Java architect, works with i2 Technologies in their B2B Marketplace Platform effort and business intelligence. He has over five years of OO design and development experience in Java, CORBA and messaging.

NIRMAL\_PATIL@I2.COM

MAJEED\_GHADIALY@I2.COM

# NETDIVE

[www.netdive.com](http://www.netdive.com)



## Application integration across the Internet

# SOAP and the Next Generation of XML

**T**he release of the SOAP 1.1 specification at the end of April was met by tremendous interest from the media, analysts and developers. Since IBM was a coauthor of the document (along with Microsoft, Lotus, DevelopMentor and UserLand), we were naturally pleased by the response. The SOAP4J toolkit that we put on the IBM alphaWorks Web site ([www.alphaworks.ibm.com](http://www.alphaworks.ibm.com)) two days after the release had over 6,000 downloads in its first month. This encouraged us to donate the toolkit to Apache in June. After that, SOAP (Simple Object Access Protocol) picked up Sun Microsystems as another important endorser. By my count, over two dozen SOAP articles have appeared in the press. Why has this technology been attracting all this attention?

Messaging protocols aren't new. If you want an envelope format to send your invoice from one server to another, you have a number of choices. SOAP represents a distillation of the basic features that are considered necessary to build messages using XML. It's straightforward and includes an optional header and a required message body. It's extensible in the sense that neither the header nor the body need to follow a particular schema, though you must specify the ones you use via namespaces. To avoid total chaos, the SOAP specification provides a recommended encoding style for the message body and a usage pattern for doing RPC via HTTP. The body encoding builds on the soon-to-be-recommended (I hope) W3C Schema language datatypes.

SOAP represents a real chance for the standardization of an important building block in the stack of technologies needed for B2B interchanges. You must realize, however, that it's not meant to be a total solution for the XML infrastructure required for e-business. We must address security issues – encryption, authorization and non-repudiation. The XML security experts at the IBM Tokyo Research Laboratories

are investigating these and other issues, and have shared many of their ideas on the SOAP mailing lists.

Furthermore, we must deal early on with reliable messaging so we know messages get delivered – and get delivered only once. The W3C recognized these issues in their response to the SOAP submission. The work of the ebXML initiative ([www.ebxml.org](http://www.ebxml.org)) covers just such requirements for messaging; I hope they have an active role in the further development of SOAP.

As I write this in mid-July, the W3C hasn't announced what they'll do with the submission. Assuming they move ahead, there'll likely be a name change since SOAP will only be the starting point for their work. We'll also have to wait to see how the final standard differs from the original submission. Nevertheless, I think the SOAP 1.1 specification is enough to start building implementations. Get the code from Apache ([xml.apache.org](http://xml.apache.org)) to start enabling your clients and Web application servers.

### How SOAP Will Enable Web Services

I expect SOAP will be widely used for transporting XML messages, and RPC

via SOAP will be part of the next important model for bringing some order to the explosive growth of B2B. At IBM we've referred to this as *service oriented architecture* or, more recently, *web services*.

One of the ideas behind Web services is to expose the functionality of your commercial or enterprise applications by providing XML APIs. For example, suppose you run a B2C retail company and want to provide a way for customers to tell if their packages have been delivered. You could provide customers with the name of the shipping company and the tracking number, then send them off to the delivery company's Web site to look up the information. However, if the delivery company exposed the "what is the delivery status" functionality as a service, you could send the company a SOAP message asking for the status. This and the response would be an XML message. You could then incorporate the information into your database and onto the Web page for your customers. You've now provided important information and kept your customers on your site. The communication with the delivery company was invisible to the customers.

If your delivery company is doing a good job, it'll likely move to a larger, more sophisticated computer operation. Because you're using SOAP and XML messages, this scaling up won't require any changes to your application. Conversely, if the company is providing poor service, you may decide to switch.

### AUTHOR BIO

Bob Sutor is IBM's program director for XML technology and the chairman of the OASIS board of directors. In his IBM role he drives integrated strategy, technology and marketing plans for XML, as well as supporting open standards activities and technical partnerships in the industry.



**I expect SOAP will be widely used for transporting XML messages, and RPC via SOAP will be part of the next important model for bringing some order to the explosive growth of B2B**



If the new company is using the same “what is the delivery status” service definition, change only the URL used to retrieve that data. Otherwise your application remains the same.

To invoke or create other services to complete your e-commerce solution, you may do catalog look-ups, inventory verification and credit card purchase validations. If SOAP and XML messages are used for this, programmers need to know fewer technologies to get their jobs done. Their skills are more useful, and so are they! The more we simplify and standardize, the easier it'll be to create sophisticated applications and add value to what we already give our customers.

Note that we didn't need to talk about programming languages or environments in any of the above. You may use Java to write your server-side e-commerce application, while the delivery company might use CORBA or DCOM. It doesn't matter because all you want is to ask a question and get a response. The world isn't going to be pure Java anymore, just as it will never be pure CORBA or DCOM. These immensely powerful and useful technologies will grow in use, but XML messaging is the way to bridge the gap between them. SOAP isn't meant to replace true distributed object systems. It can connect such systems when the full richness and complexity isn't needed or isn't practical across the Internet.

### Industry Standards: The Next Round

In the e-commerce delivery example I mentioned changing shipping companies if the quality of service dropped. Changes could also occur if you get a better price elsewhere or simply add another company to handle all your packages. You can switch or add companies without drastically recoding your application if they use the same service descriptions. Clearly, this is to your advantage. It's also to their advantage if they can entice you to start using them with a minimum of fuss.

I believe that agreement around service descriptions in XML will be the next phase of e-business industry standardization. Just as the auto, insurance, health care, pharmaceutical and other industries are actively creating XML standards, I think they'll come together to provide common service descriptions. How will they be specified? In May Microsoft released its Service Description Language (SDL), and in June IBM introduced the Network Accessible Services Specification Language (NASSL). There's hope for a convergence of these two and subsequent official standardizations. Once that begins to happen, the vertical industries can get involved and move B2B forward in a more orderly manner.

As an exercise, I encourage you to think about the services you might provide as gateways into your applications. Think about how you might use SOAP for application integration across the Internet. I think we'll find that enabling enterprise legacy applications with services' front ends will prove to be both a practical and lucrative activity for companies large and small. ☎



[ WRITTEN BY MARK WARDELL ]

As software developers, you may be receiving requests for XML support, implementing XML support or using a third-party package that supports or will soon support XML data interchange. Two good reasons to support XML are:

1. Browser and development tool vendors have announced their XML plans in full or in part.
2. XML promises to deliver better interoperability between applications and across all platforms – local and remote.

This article presents XML in the context of data interchange, a text file format initially introduced as a document authoring tool or a better HTML. Many developers are now deciding that XML is a great format for electronic data interchange.

XML data interchange is easy. In this article I'll demonstrate the addition of XML support to a preexisting application – a clone of the Minesweeper game from the Windows family of operating systems. I originally wrote the article while learning the Java Swing class libraries and grid layouts. I won't discuss those topics but will focus only on the addition of XML support. This application links with the IBM XML4J and the Sun JAXP1.0 libraries, and assumes JDK 1.2 or higher.

Early adopters of any new technology expose themselves to risks. I'll identify these risks and a roadmap for minimizing your exposure. The approach is through the tried-and-true object-oriented method of encapsulation. We can also minimize exposure by deciding what parts of XML to embrace and what to temporarily ignore. Let's look at the existing standards.

### XML Data Standards and Parsers

XML is standards based. The document itself is standardized, as are the programming libraries that read and write XML data. An XML text file can have a separate related file called a DTD (Document Type Definition) that tells an XML-enabled application what to look for when validating XML data. Since the DTD is controversial, alternate XML schemas have been proposed. These were still under development at the time of this writing. Due to this change you may want to consider *DTDless parsing*.

XML is read and written with a parser. At this stage parsers come in two somewhat standardized flavors: DOM (Document Object Model) and SAX (Simple API for XML). In October 1998 the W3 Consortium defined the DOM parser. The SAX was defined later outside the W3C, mostly as a result of the dissatisfaction with the DOM model. The consortium went as far as to define Java parser interfaces. However, they didn't define the actual classes for parsing, just the Java interfaces. The following is from the Sun Web site:

"However, the Level 1 DOM specification is silent on the subject of how to input and output. It tells you how a DOM has to operate, but does not cover methods for...reading or writing XML. As a result, you can't create a DOM from an existing XML file without going outside the DOM Level 1 specification."

Potentially a vendor may provide an input method, but not an output one, and maintain DOM Level 1 compliance. Since the parser libraries for DOM are defined only in terms of required interfaces, you can't code the import statement at the top of your Java classes with any accuracy, because every vendor has different classes that implement the DOM

# GETTING STARTED WITH XML DATA INTERCHANGE



# Choosing a vendor-neutral approach to maximize flexibility

Level 1 standardized Java interfaces. The approach taken here, and demonstrated in the sample code, emphasizes encapsulation of any vendor's Java class. This is appropriate, especially in light of the schema/DTD debate that exists over the data format. Different vendors' parsers will support schemas at different times, or not at all. If you need an XML schema in the future, it would be nice if your application's code could minimize the impact of changing the parser library.

## DOM and SAX Compared

Both the Sun and IBM libraries provide DOM and SAX parsing strategies. DOM parsers require the entire document to be "well formed" or parsable before it can be read. If any of the XML is bad, the DOM parser won't receive any data. This isn't true of SAX, which parses only a portion of the data at a time. SAX will raise an exception for bad data portions. If your data is large, consider using a SAX parser or chaining together smaller DOM trees with user-defined continuation markers in the data itself. The DOM model provides an in-memory tree representation of your data. The SAX model triggers an end element event as it parses the XML and allows you to create your own data structures at that time. DOM was first and is more formally standardized by the W3C standards body. The SAX model, less rigorously standardized, is supported by many parser vendors. Be aware that any data read by a DOM parser can be read by SAX. The reverse, however, isn't always true. Remember, DOM parsing is all or nothing, while SAX implements error tolerance, throws an exception on a bad XML section and continues parsing. The sample code uses the DOM model mostly for ease of use. I like the in-memory tree for small applications. For developers concerned about program size and algorithm efficiency, further investigation of SAX is warranted.

## Getting Started

To build in flexible support for XML consider the following:

- **DTDless parsing:** If you're interoperating with another XML-enabled application, it may require a DTD; therefore DTD-less parsing may not be an option. If you're able to parse DTD-less, your data interchange strategy will have a greater choice of parsers. Also, memory may be conserved. Doing DTD-less parsing in the data interchange context is analogous to writing to a database with no constraints defined. However, if you do succeed at an initial DTD-less approach, you'll have more options when or if schema-enabled parsers become available. The drawback is that the code has to be written to ensure that valid values are placed in and taken from the XML stream.
- **Use as little of any vendor's library as is feasible:** If you look at Figure 1 and Table 1, you can see that XML parsing libraries are large – IBM has over 150 classes. There are considerable differences in the number of classes and interfaces each vendor provides in the parsing libraries. The DOM parser is supported by the same 18 files in each library. If you do code to the superset, changing libraries later will be cumbersome.

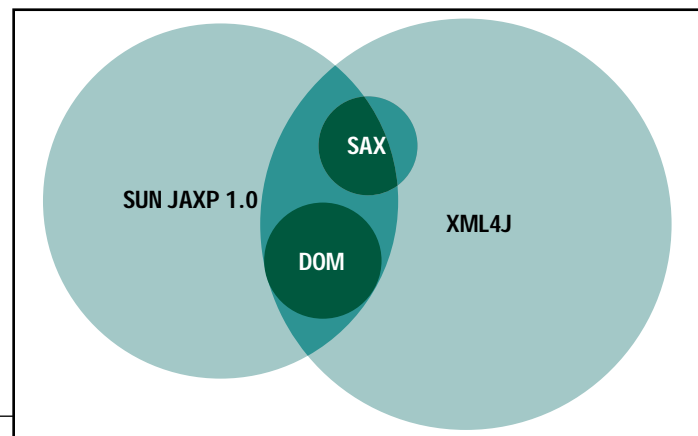


FIGURE 1 Parsing libraries

Package	IBM #of java files	Sun # of files
org/w3c/dom/*.java	18 (all interfaces)	18 (all interfaces)
org/xml/sax/*.java	13 + /helpers +misc	10
All java files	150+	80+

TABLE 1 Parsing Libraries

- **Use the standards-based section in an abstract base class:** Here we simply take the standards-based DOM parser because DOM's interface specifications are identical. *Note:* SAX has some files in XML4J that aren't in the JAXP1.0 library.
- **Implement vendor specifics and I/O in base class descendants:** Because DOM requires that only interfaces be declared, it's up to the vendor to implement them in a class. Since these class names aren't standardized, further knowledge of the vendor's library is required. I've put this knowledge in descendants of the abstract base class.

## Class Framework Overview

A quick overview of the software layers may be helpful. If your organization is designing applications correctly, you'll find clear divisions between the persistence, business logic and presentation layers. Figure 2 provides an overview of the sample code's division of layers. The sample code had no preexisting persistence layer. It was added only for demonstration purposes in this article. Extensive use of the Observable model is used. Any class implementing the Observer interface in Java can add themselves as an Observer of a descendant of an Observable class. In our example code in GameLogic's constructor a call is made to fGetterSetter.AddObserver(this). Subsequently, when ParserWrapper calls setChanged() and notifyObservers(), the update() method of GameLogic is called. This Observer/Observable pattern is also used to communicate between the nonvisual layer – GameLogic and the Java application's mainframe.

Fundamentally, the object model subsumes the lower layers via "has-a" relationships or containment. On the presentation layer the JFrame descendant "has-a" GameLogic instance, which "has-a" ParserWrapper instance. The GameLogic instance uses late binding to deter-

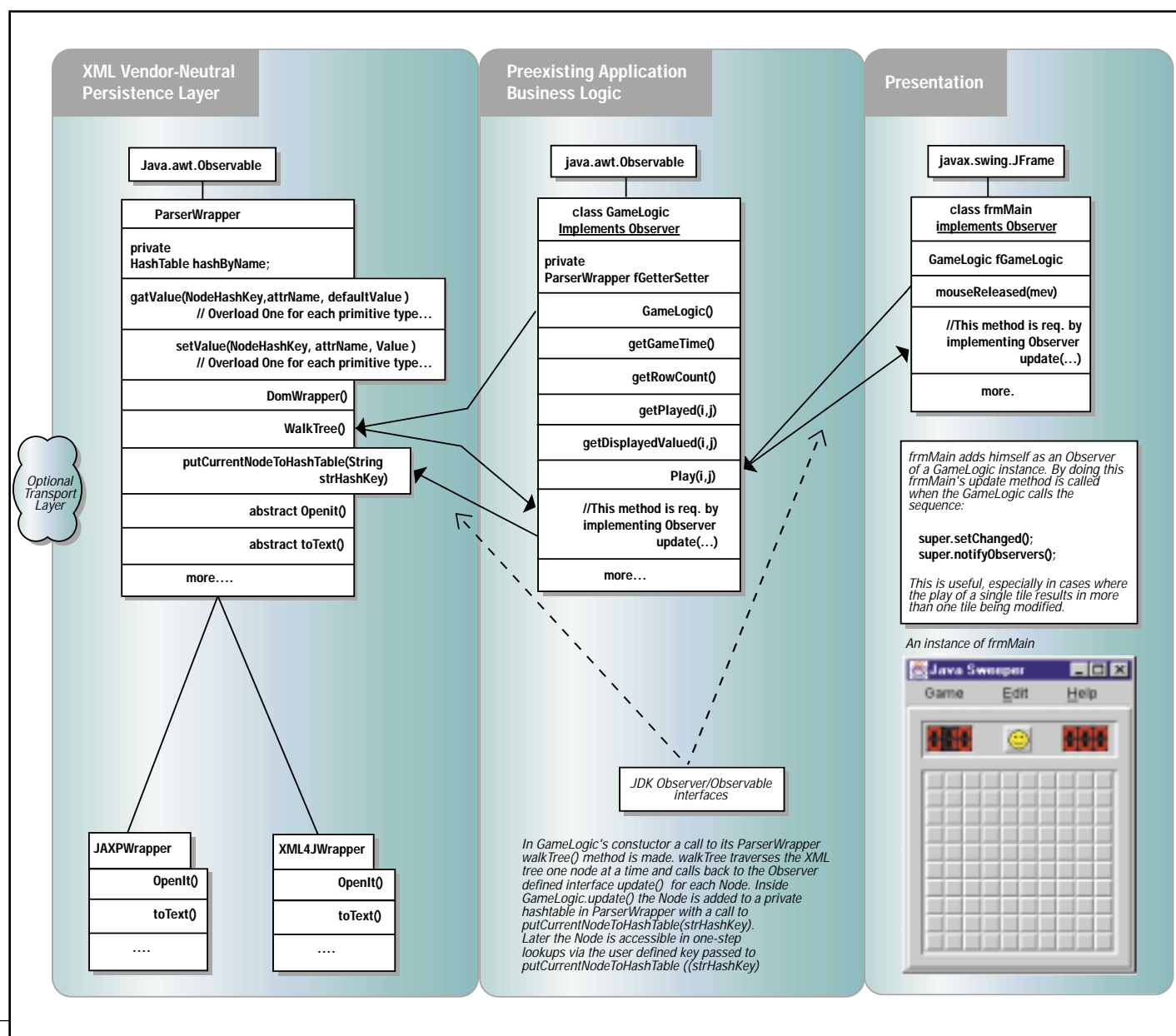


FIGURE 2 Division of persistence, business logic and presentation layers

mine if it should instantiate a Sun or IBM parser object. You could choose a more Model-View-Controller-compliant object hierarchy, but the simple containment model is sufficient to demonstrate an XML vendor-neutral object hierarchy. The model is event-driven through the Observer/Observable technique of the JDK. *Note:* In this Internet-enabled age there'll most likely be a separate transport layer to the left of the persistence layer.

## The Data

The first focus of an XML project should be data modeling. If it occurs first, more teams can begin their detailed designs in the business logic and transport layers. If DTDs are used, it's the responsibility of the data modeler to provide an XML file sample and a corresponding DTD. In my simple DTD-less example I modeled an XML file using Microsoft's XML Notepad beta 1.5 (see Figure 3). The game will save itself to an XML file. XML is relatively self-documenting. Define your elements as you see fit. To review the contents, open the saved game data with the XML Notepad. Note that at this time Microsoft's XML Notepad doesn't support DTDs. XML Notepad can be downloaded for free from <http://msdn.microsoft.com/xml/NOTEPAD/download.asp>. XML schemas and DTDs allow the data modeler to define the constraints of data without specifying the code, similar to constraints on a database. Reasons to use them will be compelling when matching standards and parsers are available. The development team with the most flexible implementation can be the first to use the new parsers.

Unless you're interoperating with another vendor, you have complete control over the data. A tool such as XML Notepad is ideal for modeling your data structures. XML files are fairly self-documenting. Figure 3 shows an XML file for a game that's been in progress for four seconds; it has 10 rows and columns. The Items element has Row elements with Piece elements that have attributes of Id (the column), ShownValue, UnderlyingValue and isPlayed. A DTD would be useful to limit isPlayed to [0,1] or ['T','F'].

## The Class

For the sample code I've chosen a DOM parser. To begin parsing it's not necessary to understand all 150-plus Java classes and interfaces provided by IBM, nor is it necessary to understand the 80-plus classes provided by Sun. Most of what you need to know can be found in five central interfaces: Attr, Document, Node, NodeList and NamedNodeMap (see Figure 4). Since these are in the DOM package, they're found in both libraries. Unfortunately, we need to go outside the specification to implement the actual I/O and find the classes that implement these interfaces. (It's beyond the scope of this article to explain the interactions of the 100-plus classes found in IBM's XML4J, but if you study the five interfaces in Figure 4 you'll have a good idea how to get started with DOM. The methods – the core interfaces imported by ParserWrapper – have been named appropriately so you should have a good idea what they do.)

It's easy to get started with these interfaces. Listing 1 provides the code that would read and write the gameTime parameter to or from an in-memory DOM tree, assuming that Node *n* is the Parent-labeled game in Figure 3.

The ParserWrapper Class provides the following:

1. Retrieves any given Node in a one-step, straightforward manner.
2. Allows user-defined keys to be added for the retrieval of any Node element on the tree.
3. Encapsulates the Node interface.
4. Implements primitive-type getters and setters that allow users to get and set typed values to the tree without type casting or translation.
5. Provides abstract definitions of the required I/O classes so implementation may leverage vendor specifics if provided.
6. Provides for SAX-like, end-element events. (This will aid encapsulation if the object ever implements SAX internally.)
7. Mapping of any data structure. (Our example most closely resembles an array, but the class can just as easily map any arbitrary data structure.)

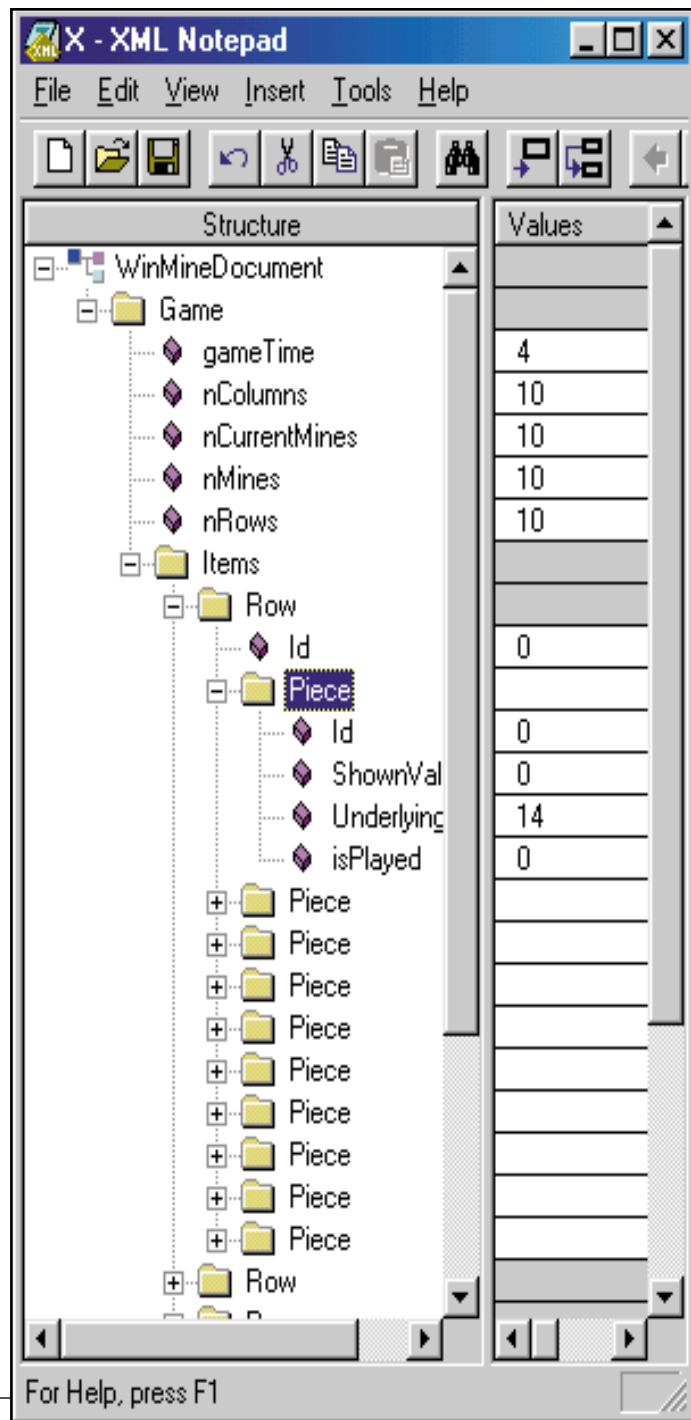


FIGURE 3 Sample output file from the Minesweeper clone game

The problem with the setGameTime() and getGameTime() samples is that the interaction with the business logic layer requires a node parameter. The Node class is a DOM interface. This object framework should be flexible enough to conceivably switch to SAX later. In addition, the business logic shouldn't have any knowledge that XML is involved in the persistence layer.

The solution is found by allowing the ParserWrapper to call back to the GameLogic class as it walks the tree. This happens when the ParserWrapper.walkTree(..) method is called. It calls back for each Node on the tree as it's traversed. The GameLogic, in this callback, can then name the Node (see the ParserWrapper method putCurrentNodeToHashTable(String strHashKey)) for later retrieval from a private hashtable in the

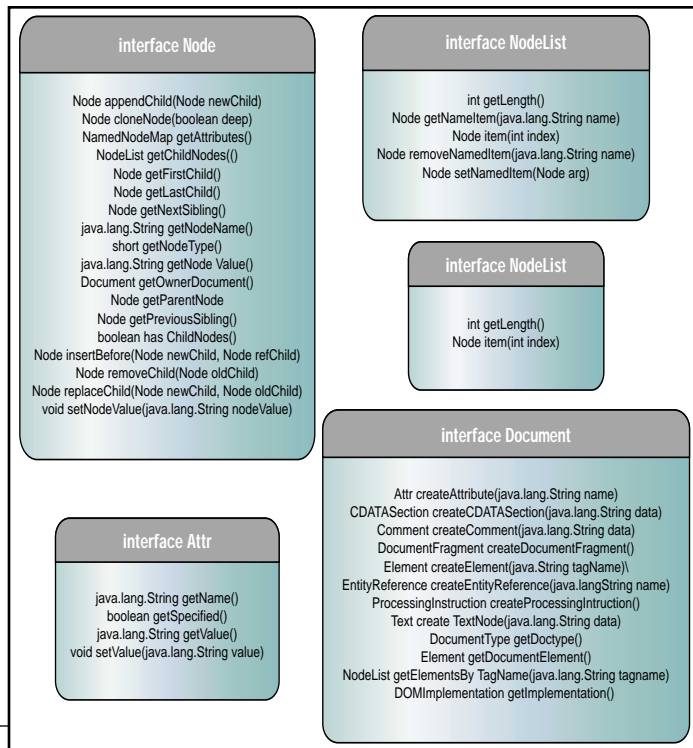


FIGURE 4 Core DOM classes

ParserWrapper Object using the `getValue(String hashKey, String attrName)` function call. At this point all XML knowledge is contained in the `ParserWrapper()` class and any attribute value can be accessed in a quick one-step fashion via user-defined hashkeys.

The callback mechanisms described above are implemented with the JDK Observer/Observable design pattern. This could be a problem if you have a predefined persistence layer that's a descendant of any class other than `java.lang.Object`. If this is the case, consider implementing your framework with a Model-View-Controller design pattern.

During construction of the `GameLogic` object, the following line of code defines the game logic as a receiver of the Sax-like events.

```

public GameLogic(ParserWrapper AGetterSetter, String FileName)
{
    fGetterSetter = AGetterSetter;
    fGetterSetter.addObserver(this); // ParserWrapper will call
    my update when it walks the tree
    fGetterSetter.openIt(FileName);
    fGetterSetter.walkTree("Game", "Items", true, false,
    false, false, false, true);
    this.setTotalMines(this.getTotalMines());
}

```

When `walkTree()` is called, since the `GameLogic` Object has added itself as an Observer, the `ParserWrapper` calls the `update()` method of the game logic:

```

private String getHashCodeKey(int I, int J){
    return "Piece["+I+"]["+J+"]";
}

public void update(Observable o, Object arg) {
    if (o.equals(fGetterSetter)){
        if (arg.equals("Row"))
            currRow = fGetterSetter.getCurrentValue("Id", 0);
        else if (arg.equals("Piece")){

```

```

            int iCol = fGetterSetter.getCurrentValue("Id", 0);
            fGetterSetter.putCurrentNodeToHashTable(this.s.getHash-
            CodeKey(currRow, iCol));
        }
    }
}

```

When the game logic needs to know if a piece is already played, it's accessed simply by the `GameLogic` objects public method:

```

public boolean getPlayed(int i, int j){
    return
    fGetterSetter.getValue(this.s.getHashCodeKey(i, j), "isPlayed", true);
}

```

## Selection Criteria

Parser selection criteria vary from application to application. On the Internet program size is always an issue. Parsers can be large – 500KB. In addition, application developers might opt to parse their XML streams with both SAX and DOM strategies in the same application. There are many issues associated with parser conformance to XML specifications. A good place to learn about them is [www.oasis-open.org/](http://www.oasis-open.org/). Oasis publishes conformance tests on their Web site.

When schema support becomes available, development teams that have chosen a vendor-neutral path that maximizes flexibility will be rewarded. Development teams should accommodate change by double-checking the flexibility of their implementations. Essentially, development teams should ask: "What is our vendor neutrality strategy?"

An interesting site on the Web that promises to help developers with vendor neutrality and parser complexity can be found at [www.jdom.org](http://www.jdom.org). JDOM believes simpler is better.

## About the Sample Code

The code presented here isn't intended as a template for the next "killer" XML app, but there are some good design elements. It would be interesting to see a third vendor's DOM parser pushed into this object model. Alternatively, you could try using Sun and/or IBM's SAX model implementations and draw comparisons. ☒

## AUTHOR BIO

Mark Wardell is a consultant for CGI in Houston, Texas. His experience includes process control, multimedia and commodities trading. Mark holds a BS from the University of Houston, Clear Lake.

MWARD@YAHOO.COM

### LISTING 1

```

public String getGameTime(Node n){
    // n is the parent Node that has a child attribute named
    "gameTime"
    NamedNodeMap attrs = n.getAttributes();
    return attrs.getNamedItem("gameTime").getNodeValue();
}

public void setGameTime(Node n, String strGameTime){
    // n is the parent Node that has a child attribute named
    "gameTime"
    NamedNodeMap attrs = n.getAttributes();
    attrs.getNamedItem("gameTime").setNodeValue(strGame-
    Time);
}

```





needed for sharing vocabularies. These repositories serve as data stores for DTDs and schemas, XML-based directory mechanisms, database structures, UML modeling tools, glossaries for relationships, context-specific terms and so on. These repositories are usually owned by consortia of industry verticals that hash out things like the meaning of an “SKU” or the elements of an “invoice.” Repositories will eventually contain standardized business components, tags, and industry terms and definitions.

XML registries, like repositories, contain common information for industries. However, they’re mainly stores for XML schemas and DTDs. The idea behind registries is that different industry representatives can submit XML schemas. Later, when some other party is looking for a similar schema, they should be able to find one they can use directly or extend. This is an excellent example of the power of the “X” in XML, which stands for “extensible.” Currently the XML industry has two main organizations that offer registries for XML schemas – the BizTalk registry from Microsoft and the recently announced XML.ORG Registry from OASIS, a consortium that consists of several organizations. The registry was formed with resources donated by Sun, IBM, Oracle, Documentum and DataChannel. It seems that the XML schemas will be split across two camps again – Microsoft and the rest of the world. At the same time other XML repositories and registries are appearing on the horizon. Obvious problems of redundancy and complexity will have to be resolved between these registries.

At XML DevCon I spoke with representatives from both Microsoft and OASIS. Microsoft has the more mature registry as it has been functional for several months now. OASIS announced theirs in June at XML DevCon. While each organization acknowledges the other’s presence, neither seems to have given much thought to interoperability between schemas registered in either one of the registries. Similar or redundant schemas may be registered in these registries, and when someone

searches for a schema, he or she will have to go across both registries. As the number of organizations offering registration services increases, this problem will be compounded. Now may be a good time for someone to create Internet directory services that can span multiple registries. Hopefully the owners of the existing repositories and registries will eventually offer cross-access capabilities over the Internet.

The other point of concern is that the registry owners aren’t very discriminating about who submits the schemas and which ones get accepted. The submittal process is fairly well defined, but almost any organization can submit a schema. The BizTalk Registry already represents schemas from 150 organizations and XML.ORG has 20 so far. By the time you read this the number will undoubtedly have increased tremendously. Proprietary schemas are also finding their way into these registries.

While this article highlights the problems that accompany repositories, I’m in favor of having standard repositories in the industry. I’m glad such efforts are taking place as they’ll reduce the chaos that inevitably emerges from open standards like XML. However, these efforts will have to consolidate at some point so the developer and business community benefit from true cross-enterprise open standard XML schemas.

### XML-J Goes Monthly

With this issue **XML-J** will be published monthly; XML is evolving so rapidly that we felt you should get valuable information as soon as possible. This change is also the result of reader support and appreciation. We hope to continue to satisfy your needs in the field of XML and related technologies. This month we have several articles that focus on XML protocols and messaging. Bob Sutor and Simeon Simonov introduce you to the rationale behind SOAP in their respective columns. Nirmal Patil and Majeed Ghadialy offer their insight on the synergy between XML messaging and JMS. Mark Wardell provides an introduction to XML. Seit-Leng Lai discusses how WML can be used to access remote devices.

Enjoy the issue, and send us your feedback – good and bad. ☎

# IXIASOFT

## www.ixiasoft.com

[ WRITTEN BY MARK BAKER ]

X

*ML is rapidly becoming the way applications communicate.*

*Because it isn't one language but a means to create many languages, it can play many roles in application integration and data exchange. In this article I'll explore some of the varied roles that XML can play by developing a simple middleware application that serves up data from a database.*

The application (see Listing 1) uses XML in three ways:

1. Requests from clients are encoded in XML.
2. Data sent back to a client is encoded in XML.
3. The database itself contains data marked up in XML.

The application is written in OmniMark – a 12-year-old language with a long history in the SGML community – which became a free language in May 1999. It's based on the streaming programming model pioneered by OmniMark and subsequently adopted by SGML and XML translation languages like DSSSL and XSL. Streaming is also the principle at work in the SAX parser interface.

OmniMark is alone, though, in having developed the streaming model into a full, generalized programming language. The conventional approach to writing a program is to design a data structure, populate it, manipulate it and finally serialize it to create output; that is, it's a memory-centric approach. The streaming approach involves acting on the data directly as it streams, without creating data structures – it's an I/O-centric approach.

An OmniMark program is structured as a collection of rules, with different rule types used for different purposes:

- The *process* rule fires when the application is started and contains the main processing logic.
- *Element* rules fire when OmniMark's integrated XML parser finds an element.
- *Find* rules are used to process textual data using a pattern-matching language similar in capabilities to regular expressions.

OmniMark provides an abstraction layer between program logic and data sources/destinations. All OmniMark programs operate on the program's current input and all output goes to the current output. You can attach any source or destination to current input and current output, locally or across a network. The program operates the same way on any data stream, independent of its source and destination.

## Server Programming

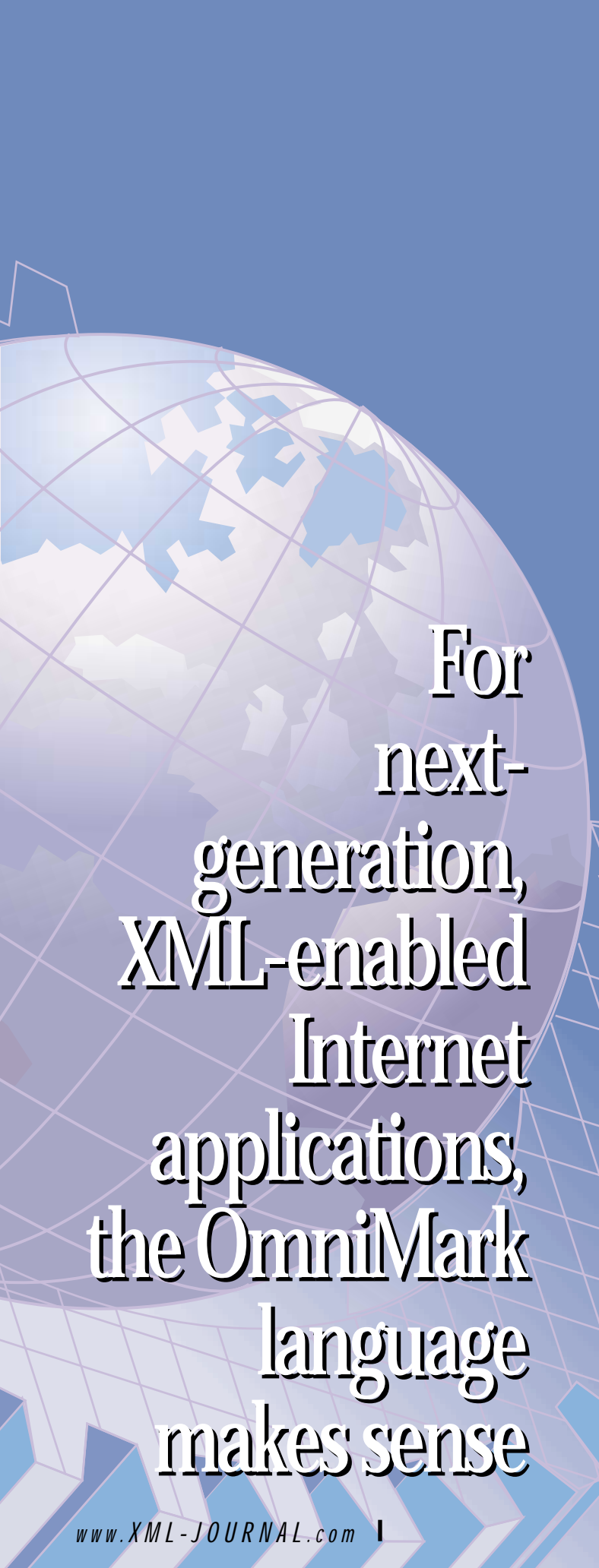
A middleware application is a server program. Its main function is to receive requests from clients and respond appropriately. What makes the server middleware is that it requests information from yet another server to satisfy its clients' requests.

The two essential performance characteristics that every server program must have are:

1. **It must survive errors:** If something goes wrong in processing a request, the server can't shut down. It must continue running so as to service the next request.

# BUILDING XML MIDDLEWARE USING OMNIMARK





# For next- generation, XML-enabled Internet applications, the OmniMark language makes sense

2. ***It must return to a stable state after each request is completed:***  
Whether or not the request was serviced successfully, the server should return to a steady state.

The basic anatomy of a server is simple:

- *The startup routine* establishes the service and any resources it needs.
- *The request service loop* receives requests and responds to them.
- *The shutdown routine* cleans up any open resources and shuts down the server.

## How to Establish a Service

OmniMark uses OMX (OmniMark extension) components to connect to external data sources. To establish a TCP service, I use the `tcpServiceOpen` open function found in the TCP/IP library. The function returns an OMX variable that's a handle to a `TCPService` OMX component that manages the TCP service. The function is called in the initializer of the "service" variable:

```
local tcpService service
  initial {tcpServiceOpen at port-number}
```

## How to Receive Requests

Once the service is established, the program must wait for a connection attempt from a client. This is accomplished with the `TCPServiceAcceptConnection` function, which takes the OMX variable representing the service and waits for a connection. When a connection is made, it returns another OMX variable, which is a handle to a `TCPConnection` OMX component that will manage the connection:

```
local tcpConnection connection
  initial {TCPServiceAcceptConnection service}
```

## Establishing the Request and Response Streams

Now that I have a TCP/IP connection, I need a way to talk to it. To output data to a TCP/IP connection, I must attach an OmniMark stream to the connection; all data written to that stream will then go to the connection. I do this with the "TCPConnectionGetOutput" function, which takes the connection OMX variable as a parameter:

```
local stream reply
...
open reply as TCPConnectionGetOutput connection
  protocol IOProtocolMultiPacket
```

The statement "protocol `IOProtocolMultiPacket`" is a second parameter to the `TCPConnectionGetOutput` function. It establishes the protocol to use for writing the data. Because a TCP/IP connection is a two-way communication channel, the sender can't signal the end of its data by closing the channel; an I/O protocol is required to establish when a message ends.

OmniMark provides support for most common protocols through the `IOProtocol` library. This program uses the `MultiPacket` protocol, which breaks up the message into packets and sends each packet prefixed by a network-long value specifying its size. The message ends with a zero-length packet.

Opening the "reply" stream with the TCP/IP connection attached creates a vehicle for sending output to the TCP/IP connection. To actually send output to the connection, I have to make "reply" the current output stream. I do this using the statement "using output as":

```
using output as reply
do
...
done
```

The "using output as" statement is a prefix to the "do" block and establishes the current output for all the code that executes within the

block, including any functions or rules called within the block. I can output data to the TCP/IP connection with a simple “output” statement anywhere within the output scope established by this statement.

To receive data from the TCP/IP connection, I use the “tcpConnectionGetSource” function. It takes the connection OMX and protocol parameters – just like “tcpConnectionGetOutput” – and returns an OmniMark source that can be used by any of the OmniMark keywords that accept sources – in this case the “scan” keyword:

```
scan tcpConnectionGetSource connection
protocol IOProtocolMultiPacket
```

## How to Survive Errors and Stay Running

The request service loop consists of a repeat loop. In OmniMark all repeat loops begin with “repeat” and end with “again.” The key to surviving an error in the course of handling a request is to catch the error in time to repeat the loop. I accomplish this by placing a single statement at the end of the request service loop:

```
catch #program-error
```

OmniMark’s catch and throw keywords provide robust structured exception handling that you can use for both flow control and error handling. A throw isn’t a GOTO. A throw starts a systematic process in which program scopes are closed one by one, starting with the scope in which the throw occurs and ending with the one in which the catch occurs. Garbage collection is automatic.

“#program-error” is a built-in catch name that I use here as the line of last defense. Any program error, any uncaught throw, any error in an external system that is not caught and handled by another catch will be caught here. The catch will result in the current iteration of the loop being shut down and tidied up. Control will then return to the top of the loop, starting a new iteration.

## How to Return to a Stable State After Each Request

The prescription for returning the server to a stable state is simple: keep all your variables local to the request service loop. That way, whether the loop ends normally or with an error, the variables will be destroyed, the garbage cleaned up and the next iteration will start with a clean slate.

At one place in this program I violate this rule. I use a global variable for the database connection (“db”), trading off a little robustness for the performance advantage of maintaining a permanent connection to the database. You can use catch and throw to detect problems with the database connection and recover from them, but that’s outside the scope of this article.

## XML Processing

OmniMark’s XML parser is fully integrated into the language. It doesn’t need or use a DOM or SAX interface.

Once parsing is initiated, the parser fires markup rules for the various markup structures it encounters. The most common of the markup rules is the *element* rule. While SAX has separate events for the start and end of an element, OmniMark fires a single element rule for each element. Each element rule uses the parse continuation operator (“%c”) to initiate parsing of the element’s content. Element rules are thus fired hierarchically. Each rule is suspended while the element’s content is parsed and resumes once the parsing of the content is complete.

Parsing is initiated by the “do xml-parse” statement:

```
do xml-parse instance
with xml-dtds {"request"}
scan tcpConnectionGetSource connection
protocol IOProtocolMultiPacket
output "%c"
done
```

The “with” clause specifies the DTD to use. In this program the request DTD is precompiled in the start-up section.

The “scan” clause specifies the source from which to read the XML document. In this case it’s the source returned by the tcpConnectionGetSource function that attaches the OmniMark source to the TCP/IP connection. What this means is that the XML document will be streamed directly from the TCP/IP connection into the XML parser.

“do xml-parse” is a block statement. Within that block the parse state has been established, but parsing isn’t actually in progress. Parsing is started by the parse-continuation operator (“%c”), which is roughly equivalent in function to the XSL statement apply-templates. “%c” is a string escape sequence that allows you to easily express where you want the content of an element to fall in the output stream.

## Parsing the Information Requests

My XML language for requests is simple. The root element is request and the request element can contain a single element representing any one of the request types. Each request-type element (“product-by-type” and so on) has the data content appropriate to the information being requested, usually a database key value. In effect, this DTD describes a simplified database query language that is specific to the particular database I’m accessing. The middleware program acts as an interpreter for that language, converting it into standard SQL queries and returning the results of those queries with XML encoding.

The parser is started in response to the “%c” in the “do xml-parse” block. When it finds the request element, it fires the element rule for “request” and pauses. The “request” element rule has no work to do, so it simply restarts the parser with “%c”.

```
element "request"
output "%c"
```

Suppose the request is for a list of selected products. The request element will contain an element, “selected-products”, whose data content will be a comma-separated list of product IDs. The element rule for “selected-products” begins like this:

```
element "selected-products"
local stream query initial
{ "SELECT ProductID, "
  || "ProductName, "
  || "ProductPrice "
  || "FROM Product "
  || "WHERE "
  || "ProductID IN (%c)"
}
```

The “%c”, which every element rule must contain, is tacked onto the end of the initializer for the variable “query”. The data content of the “selected-products” element is streamed into the variable “query” and becomes part of the SQL statement that will be used to query the database. In effect, the product line ID has been streamed directly from the TCP/IP connection into the SQL statement.

## Database Access

OmniMark uses OMX components to communicate with all external data sources, so communication with a database uses a database OMX provided by the omdb library. The connection to the database is established when the variable “db” is initialized as part of the start-up routine:

```
global dbDatabase db initial {dbOpenODBC dsn}
```

The dbOpenODBC function takes a parameter that is the data source name (DSN) used by the ODBC driver manager to identify a database. It returns a database OMX variable.

## Querying the Database

Once I have a database connection and a completed SQL query, I query the database with the dbQuery function:

# **XML BOOT CAMP**

**[www.sdexpo.com/bootcamp](http://www.sdexpo.com/bootcamp)**

```
dbQuery db sql query record rs
```

The dbQuery function takes three parameters, the OMX variable for the database, the SQL query – heralded by the word “sql” – and an OmniMark shelf named “rs” – heralded by the word “record.”

A shelf is an OmniMark data structure. It’s an associative array, meaning that items can be addressed either by position or by a textual key value. “dbField” is an OMX variable type for an OMX component representing a database field. The “dbQuery” function will populate the “rs” shelf with “dbField” OMX variables representing the fields of the current record. The names of the fields will become the keys of the shelf.

After executing the query, the program checks to see if any records were returned; if not, it throws “record-not-found”:

```
throw record-not-found unless dbRecordExists rs
```

This throw is caught by the statement:

```
catch record-not-found
  output '<response status="notfound"/>'
```

Throwing out of an element rule terminates the current parse. The code in the catch block then outputs the XML message:

```
<response status="notfound"/>
```

Since the “do XML parse” block is within the output scope created by the statement “using output as reply”, this output goes straight to the TCP/IP connection and to the client.

Figure 1 illustrates how data streams through the program. In this figure the top line shows the streaming of the request data from the TCP/IP port to the XML parser and into the SQL query. The query itself is passed as a function call to the database OMX, not streamed. The bottom line shows the streaming of the response data from the database to the find rules that escape markup characters in the text, the interpolation of the XML tagging by the program, and the streaming of the result to the TCP/IP port.

## Building an XML Encoded Response

If the query does contain records, the program outputs the “response” element with the status “ok” and then loops over the records and outputs the data surrounded by the appropriate XML tags. At the end of the loop it outputs the “response” end tag. All this happens in the output scope in which the parse was initiated, so the output all goes to the TCP/IP connection:

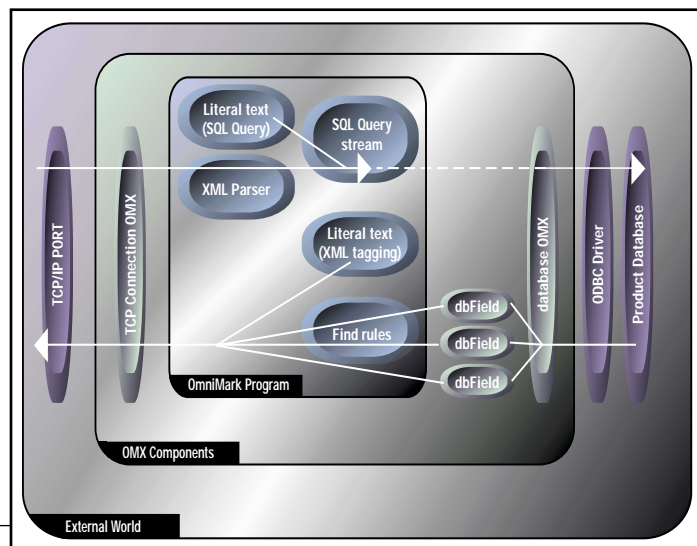


FIGURE 1 The streaming of the data in the middleware program

```
output '<response status="ok">%n'
repeat exit unless dbRecordExists current-record
  output '<product>%n<id>'
  || dbField dValue current-record{"ProductID"}
  || '</id>%n<name>'
  submit dbField dValue current-record{"ProductName"}
  output '</name>%n</product>%n'
  dbRecordMove current-record
again
output '</response>'
```

Sometimes it takes more than one SQL query to collect the information needed to construct a response. This is the case for the product-by-line and product-by-type requests, both of which return a description of the product type or product line followed by a list of products. Because there are two separate requests, either of which can fail, I buffer the output until both queries have succeeded.

To do this, I attach the stream “response-buffer” to a buffer and make it the current output scope for the duration of the two queries:

```
local stream response-buffer
open response-buffer as buffer
using output as response-buffer
do
...
done
close response-buffer
output response-buffer
```

After the block governed by “using output as response-buffer,” I close the stream and output it. The original output scope was restored when the block ended, so the output once again goes to the TCP/IP connection.

## Dealing with the Markup in the Database

The database contains XML markup in the description field. The markup is simple; a “description” element can contain paragraph (“p”) elements, which can contain text or “prodref” elements. A “prodref” is a reference to another product in the database.

I deal with this markup by simple inclusion. One of the virtues of XML is that because of its linear nature and nested structure, the root element of one XML document can become an element in another document simply by dropping it in place. Because I control both the database and the server, I don’t have to worry about namespace conflicts. In effect, the “description” language used in the database is just a subset of the “response” language used by the server.

## Escaping the Markup Characters in the Data

The other database fields contain plain text data. Whenever you create XML from text data, you must escape the markup characters “<,” “>” and “&” to prevent their being mistaken for markup by the parser receiving the XML. Naturally, OmniMark handles this in a streaming manner.

Rather than outputting the values of the fields directly, the program “submits” them. Submitted data is processed by find rules, which apply pattern-matching techniques to data streams. (The “dbFieldValue” function returns an OmniMark source, not a string, so the data is being streamed, not copied here.)

OmniMark supports a full pattern-matching language. However, this program requires only literal text matching. Here’s the find rule for escaping the “<” character:

```
find "<"
  output "&lt;";
```

This find rule looks for “<” in the streaming data. When it finds it, it removes the matched character from the stream and outputs the escape sequence “&lt;” in its place. All the find rules are active at once so all the

```
|| product-id
|| '</product></request>'
```

## Processing the Response


The response is streamed directly into the parser, just as in the server program. The only difference in the client is that the DTD (see Listing 3) is fed to the parser as text rather than being precompiled.

```
do xml-parse document
  scan file dtd-file-name
  || TCPConnectionGetSource connection
  protocol IOProtocol MultiPacket
  output "%c"
done
```

The response is fed to the parser, which fires element rules as before. The element rules output simple HTML tagging in place of the XML tagging in the response.

## Summary

The greatest virtue of XML is that it exploits the simplicity and universality of the common linear text stream. Streams are easy to create and easy to transmit; adding XML makes streams easy to interpret. This allows XML to serve many purposes in communication between applications. It allows you to build applications that are simple, elegant and easy to maintain.

Because it combines broad-based connectivity, a streaming programming model and an integrated parser, OmniMark is a good language for building the next generation of XML-enabled Internet applications. 

## XML Resource

*OmniMark Technologies:* [www.omnimark.com](http://www.omnimark.com)

## ■ AUTHOR BIO

*Mark Baker is senior technical communicator for OmniMark Technologies. He is a coauthor of HTML4 Unleashed (second edition, 1998), for which he provided the XML chapters, and author of the forthcoming Internet Programming with OmniMark.*

```

;xml m d l w r . x o m
; a m i d d l e w a r e a p p l i c a t i o n t o a c c e s s
; a p r o d u c t d a t a b a s e a n d r e t u r n
; t h e r e s u l t s i n X M L

; i n c l u d e l i b r a r y f i l e s
i n c l u d e " o m d b . x i n "
i n c l u d e " o m t c p . x i n "

; g l o b a l v a r i a b l e s
g l o b a l c o u n t e r p o r t - n u m b e r i n i t i a l
{ 5 4 3 6 }
g l o b a l s t r e a m d s n i n i t i a l { " g o l d " }
g l o b a l s t r e a m p o i s o n - p i l l i n i t i a l
{ " _ d i e . " }
g l o b a l d b D a t a b a s e d b i n i t i a l { d b O p e n O D B C
d s n }
g l o b a l s t r e a m r e q u e s t - d t d i n i t i a l
{ " < ! d o c t y p e r e q u e s t [ "
| | " < ! e l e m e n t r e q u e s t ( p r o d u c t "
| | " | l i s t - o f - l i n e s "
| | " | l i s t - o f - t y p e s "
| | " | p r o d u c t s - b y -
t y p e "
| | " | p r o d u c t s - b y -
l i n e "

```

```

initial {tcpServiceOpen at port-
number}

throw shut-down
  when tcpServiceIsInError service

;server start up sequence

;compile the request dtd
do xml-parse document
  creating xml -dtds{"request"}
  scan request-dtd
  suppress
done

;set db to dbOpenODBC dsn

;request service loop
repeat
  local tcpConnection connection
  initial {TCPServiceAcceptConnec-
tion service}
  local stream reply

  open reply
  as TCPConnectionGetOutput con-
nection
  protocol IOProtocol Multi Packet

```



```

using output as reply
do xml-parse instance
with xml-dtds{"request"}
scan tcpConnectionGetSource
connection
protocol IOProtocol MultiPack-
et
output "%c"
catch bad-request
output '<response
status="badrequest">'
catch record-not-found
output '<response
status="notfound"/>'
catch #program-error
output '<response
status="error"/>'
done
catch #program-error
again
; shutdown
catch shut-down
output "Shutting down.%n"
; element rules for handling requests
element "request"
output "%c"
element "product"
local dbField rs variable
local stream query
initial
{ "SELECT Product.ProductID, "
|| "Product.ProductName, "
|| "Product.ProductLineID, "
|| "ProductLine.ProductLineName, "
|| "ProductType.ProductTypeName, "
|| "Product.ProductDescription, "
|| "Product.ProductPrice "
|| "FROM (Product LEFT JOIN "
|| "ProductLine "
|| "ON Product.ProductLineID = "
|| "ProductLine.ProductLineID) "
|| "LEFT JOIN ProductType "
|| "ON Product.ProductTypeID = "
|| "ProductType.ProductTypeID "
|| "WHERE ProductID=%c"
}
dbQuery db sql query record rs
throw record-not-found
unless dbRecordExists rs
output '<response status="ok">%n'
|| '<product>%n<id>'
|| dbField dValue rs{"ProductID"}
|| '</id>%n<name>'
submit dbField dValue rs{"ProductName"}
output '</name>%n<line id="'
|| dbField dValue rs{"Product-
LineID"}
|| '" name="'
submit dbField dValue rs{"ProductLine-
Name"}
output '" />%n<type id="'
|| dbField dValue rs{"ProductType-
ID"}
|| '" name="'
submit dbField dValue rs{"ProductType-
Name"}
output '" />%n'
|| dbField dValue rs{"ProductDe-
scription"}
|| '%n<price>'
|| dbField dValue rs{"Product-
Price"}
||

```

```

'</price>%n</product>%n</response>'
element "list-of-lines"
local dbField rs variable
local stream query
initial
{ "SELECT ProductLineID, "
|| "ProductLineName, "
|| "ProductLineDescription "
|| "FROM ProductLine"
}
dbQuery db sql query record rs
throw record-not-found unless
dbRecordExists rs
output '<response status="ok">%n'
|| '<list-of-lines>%n'
repeat exit unless dbRecordExists rs
output '<line>%n<id>'
|| dbField dValue rs{"Product-
LineID"}
|| '</id>%n<name>'
submit dbField dValue rs{"Product-
LineName"}
output '</name>%n'
|| dbField dValue
rs{"ProductLineDescrip-
tion"}
|| '</line>%n'
dbRecordMove rs
again
output '</list-of-lines></response>'
suppress
element "list-of-types"
local dbField rs variable
local stream query initial
{ "SELECT ProductTypeID, "
|| "ProductTypeName, "
|| "ProductTypeDescription "
|| "FROM ProductType"
}
dbQuery db sql query record rs
throw record-not-found unless
dbRecordExists rs
output '<response status="ok">%n'
|| '<list-of-types>%n'
repeat exit unless dbRecordExists rs
output '<type>%n<id>'
|| dbField dValue rs{"Product-
TypeID"}
|| '</id>%n<name>'
submit dbField dValue rs{"Product-
TypeName"}
output '</name>%n'
|| dbField dValue
rs{"ProductTypeDescrip-
tion"}
|| '</type>%n'
dbRecordMove rs
again
output '</list-of-types></response>'
suppress
element "products-by-type"
local dbField rs variable
; capture the ID since we will need
to use
; it twice
local stream response-buffer
local stream id initial {"%c"}
local stream query initial
{ "SELECT ProductTypeID, "
|| "ProductTypeName, "
|| "ProductTypeDescription "
|| "FROM ProductType "

```

```

|| "WHERE ProductTypeID=%g(id)"
}
; buffer the response in case of
error
open response-buffer as buffer
using output as response-buffer
do
output '<response status="ok">%n'
|| '<products-by-type>%n'
; first get information on the
product line:
dbQuery db sql query record rs
throw record-not-found
unless dbRecordExists rs
output '<type><id>'
|| dbField dValue rs{"Product-
TypeID"}
|| '</id><name>'
submit dbField dValue rs{"Product-
TypeName"}
output '</name>'
|| dbField dValue
rs{"ProductTypeDescrip-
tion"}
|| '</type>'
; then get information on the
; products of that type
output-products where "Product-
LineID=%g(id)"
output '</products-by-
type></response>'
done
close response-buffer
output response-buffer
element "products-by-line"
local dbField rs variable
; capture the ID since we will
; need to use it twice
local stream id initial {"%c"}
local stream response-buffer
local stream query initial
{ "SELECT ProductLineID, "
|| "ProductLineName, "
|| "ProductLineDescription "
|| "FROM ProductLine "
|| "WHERE ProductLineID=%g(id)"
}
; buffer the response in case of
error
open response-buffer as buffer
using output as response-buffer
do
output '<response status="ok">%n'
|| '<products-by-line>%n'
; first get information on the
product line:
dbQuery db sql query record rs
throw record-not-found
unless dbRecordExists rs
output '<line><id>'
|| dbField dValue rs{"Product-
LineID"}
|| '</id><name>'
submit dbField dValue rs{"Product-
LineName"}
output '</name>'
|| dbField dValue
rs{"ProductLineDescrip-
tion"}
|| '</line>'
; then get information on the
; products in that line
output-products where "Product-
LineID=%g(id)"

```



```

        output '</products-by-
line></response>'
    done
    close response-buffer
    output response-buffer

    element "selected-products"
        local dbField rs variable
        local stream query initial
        {
            "SELECT ProductID, "
            || "ProductName, "
            || "ProductPrice "
            || "FROM Product "
            || "WHERE "
            || "ProductID IN (%c)"
        }

        dbQuery db sql query record rs
        throw record-not-found unless
        dbRecordExists rs

        output '<response status="ok">%n'
        || '<selected-products>%n'
        repeat exit unless dbRecordExists rs
        output '<product>%n<id>'
        || dbField rs{"Product-
tID"}
        || '</id>%n<name>'
        submit dbField rs{"Product-
Name"}
        output '</name>%n<price>'
        || dbField rs{"Product-
Price"}
        || '</price></product>%n'
        dbRecordMove rs
        again
        output '</selected-
products></response>'

    element "die"
        ;check that the die request has
        ;the proper poison-pill
        throw shut-down when "%c" = poison-
pill
        ;otherwise just ignore the request

    markup-error
        throw bad-request

    ;find rules for escaping text in XML
    documents
    find "<"
        output "&lt;"

    find ">"
        output "&gt;"

    find "&"
        output "&amp;"

    find "'"
        output "&quot;"

    ;function to output list of products
    define function output-products
        where value stream where-clause
        as
        local dbField rs variable
        local stream query initial
        {
            "SELECT ProductID, "
            || "ProductName, "
            || "ProductDescription, "
            || "ProductPrice "
            || "FROM Product "
            || "WHERE "
            || where-clause
        }

        dbQuery db sql query record rs
        throw record-not-found unless

```

```

dbRecordExists rs
    repeat exit unless dbRecordExists rs
    output '<product>%n<id>'
    || dbField rs{"Product-
tID"}
    || '</id>%n<name>'
    submit dbField rs{"Product-
Name"}
    output '</name>%n'
    || dbField rs{"ProductDescription"}
    || '<price>'
    || dbField rs{"Product-
Price"}
    || '</price></product>%n'
    dbRecordMove rs
    again

```

## LISTING 2

```

; stub.xom
; to test the product server
include "omtcp.xin"
include "ombcd.xin"

declare catch connection-error

;globals
global stream product-server-host
initial {"localhost"}
global counter product-server-port
initial {5436}
global stream product-id initial {"4"}
global stream output-file-name initial
{"out.htm"}
global stream dtd-file-name initial
{"response.dtd"}

process
    local stream output-file
    local TCPConnection connection ini-
tial
    {TCPConnectionOpen
        on product-server-host
        at product-server-port
    }

    ; check the connection was made
    throw connection-error
    when TCPConnectionIsInError connec-
tion

    ; send the request
    set TCPConnectionGetOutput connection
    protocol IOProtocolMultiPacket
    to '<request><product>'
    || product-id
    || '</product></request>'

    ; uncomment to inspect response
    ; output TCPConnectionGetSource connec-
tion

    ; protocol IOProtocolMultiPacket

    ; process the response
    open output-file as file output-
file-name
    using output as output-file
    do xml-parse document
    scan file dtd-file-name
    || TCPConnectionGetSource connec-
tion
    protocol IOProtocolMultiPacket
    output "%c"
    done

    catch connection-error
        output "Connection error%n"

```

```

; element rules
element "response"
    do scan attribute "status"
    match "ok" =|
        output "%c"
    else
        output "<H3>Server Error</H3>"
        halt
    done

element "product"
    output "%c"

element "id"
    suppress

element "name"
    output "<H3>%c</H3>%n"

element "line"
    output "%c<P>Product line: "
    || attribute "name"

element "type"
    output "%c<P>Product type: "
    || attribute "name"

element "price"
    output "<p>Price: "
    || "<$.NNZ.ZZ>" % bcd "%c"

element "description"
    output "%c"

element "p"
    output "<p>%c</p>"

element "prodref"
    output "<b>%c</b>"

```

## LISTING 3

```

<!element response (product*|line*)>
<!attlist response status CDATA
#REQUIRED>
<!element id (#PCDATA)>
<!element product (id, name, line, type,
description, price)>
<!element name (#PCDATA)>
<!element description (p+)>
<!element prodref (#PCDATA)>
<!attlist prodref id CDATA #REQUIRED>
<!element price (#PCDATA)>
<!element p (#PCDATA | prodref)*>
<!element line EMPTY>
<!attlist line id CDATA #REQUIRED
name CDATA #REQUIRED>
<!element type EMPTY>
<!attlist type id CDATA #REQUIRED
name CDATA #REQUIRED>
]>

```





*Patterns provide the recipe for the  
Java – XML provides the sweetener*

# Java, XML and the Command Pattern

**T**his month's column shows the advantages of using Java and XML to implement the Command Pattern. It also provides a brief illustration of how this pattern can be used to implement transaction integrity via compensating actions.

Patterns provide developers with reproducible solutions to common problems. The format for describing a pattern consists of a description of the problem, a brief description of the solution, the solution to the problem, good and bad consequences to the solution, and a list of related patterns.

The Command Pattern is a behavioral pattern. Behavioral patterns are used to combine, organize and manage behavior. The Command Pattern encapsulates commands as objects so they can be manipulated by a controlling subsystem that plays the role of a command manager. The command manager is capable of controlling the selection and orderly execution of commands. Historical command logs can be kept to enable the rollback of commands.

In a complementary fashion, Java provides a heterogeneous framework for deploying applications while XML provides a cross-platform framework for representing objects. Together they provide a perfect framework for implementing the Command Pattern across heterogeneous applications.

The Java framework receives commands over the Web via servlets. XML enables objects to be represented, packaged and passed across the network. By taking the common user interface extension to the Command Pattern and substituting it with XML, we're able to disjoin the command generation subsystem from specific command class implementations. This allows any heterogeneous Web-enabled application to submit XML commands to a servlet-

based system. The extension is achieved by passing the information and the name of the command inside the XML message. A command factory processes this information and is responsible for creating instances of command classes (see Figure 1). This level of abstraction allows the dynamic manipulation of command names to class-type mapping.

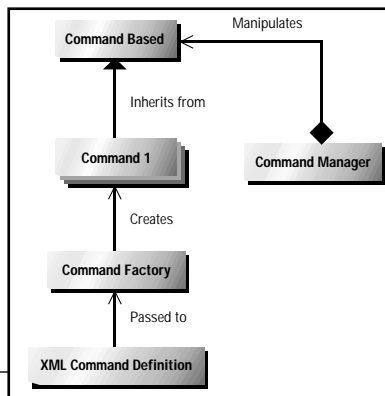


FIGURE 1 Command Pattern class hierarchy

Once the command object has been created, it's executed by the command manager (see Figure 1). The command manager is responsible for updating the history command information, processing the command execution and managing the output associated with the execution. When calling undo or rollback, the Command Manager is responsible for reading the history log and executing a compensating command to undo the state changes associated with the execution of the first command.

To understand the advantages of XML and Java when implementing the Command Pattern, let's create a scenario of a furniture store that wants to expand their services, not only over the Web but also within the store.

Nacme, a local furniture store, has decided to extend their services to support over-the-Web purchases. They're interested in providing a system that enables them to deduct merchandise from their master inventory system in real time. The inventory system provides a maximum quantity of available furniture that we can deduct from in order to fulfill an order. It supports a socket-based ASCII protocol. In addition, they're interested in having a wireless system within their store to collect customers' shopping cart information. This information needs to be automatically processed against their master inventory system.

The Web-based system they like is Java Servlet-based and the wireless system is C-based. Neither system provides a native interface to their master inventory system. Is this a problem or an opportunity? Nacme's IT department wants to minimize the amount of integration they'll have to write between the three systems. The IT department would like to treat their inventory system as a black box and send it commands to decrement or increment the inventory. It's no surprise that the Command Pattern can solve their problem. The four steps to formulating a solution are:

1. **Identify the types of commands we're interested in abstracting.** In this case they're increase and decrease.
2. **Identify the various system boundaries.** In this case both systems, the

## AUTHOR BIO

Israel Hilerio is a member of a leading e-commerce firm in Dallas, Texas, focusing on Web-based e-commerce applications and new architectures.

Web-based and the wireless-based, need to produce XML commands that will be fed to the inventory system (see Figure 2).

3. **Identify the XML command syntax.** This includes the SKU and the quantity that needs to be manipulated (see Figure 3).
4. **Identify the Java classes that will hold the command information.** The reason for migrating the information contained in XML objects to Java objects is efficiency. It's more efficient to access information and to manipulate Java objects inside a Java program than it is to manipulate XML hierarchies as objects (see Figure 4).

Another advantage to creating an XML command abstraction is that additional integrations need to support only the generation of XML commands. Additional system interfaces can range from Palm Pilots to cell phones. Java was selected as the implementation platform because it's capable of running on both UNIX and NT. As Nacme grows, they see themselves migrating hardware platforms to what's considered more mission-critical UNIX systems. Since their master inventory system software runs on NT and UNIX, they believe the Java platform will allow them to preserve their investment in their integration solution.

Another advantage of this architecture is that the interfaces, Web and wireless, complement each other and allow customers to enhance their shopping experience. They can order a product online, then cancel it by calling the store. In a similar fashion, customers can order a product at the store, then cancel online. The main complexity associated with this type of flexibility is keeping track of the order information. This can be done as part of the command manager functionality. As commands are processed by the application, they're logged into the system and associated with the submitter. This provides the security authorization required to cancel an order. In addition, the system can retrieve the previous action committed by the user and execute a compensating action on behalf of the user. Compensating actions are used to return the system to its original state. In some cases the system state can't be restored exactly; however, this isn't required as long as the system can be returned to a transaction integral state. For example, a user submitted an order to purchase five additional dining chairs. If the user wants to cancel the order, the system submits a command

to increase inventory by five chairs, assuming the chairs weren't in the process of being delivered. If they were being delivered, the system won't allow the user to cancel the order. He or she has to receive the chairs and then manually submit an RMA process.

One consequence associated with the Command Pattern is that the object that triggers the execution of the command isn't the one that executes it. However, this consequence can be viewed as a positive side effect since it allows the Command Manager to collect command history and delegate command execution to other subsystems. The command history can be used by another system to replay command executions and, in some cases, replay command attacks. Perhaps a more important side effect of this pattern is that new commands can be added without affecting the dependencies and control flow of existing commands. This is critical if we want to design a system that can grow over time.

Taking this approach to the next level, imagine how it can be used and modified to provide transaction integrity on systems that support compensating transactions. The history log becomes the transaction log and is used to roll back transactions based on the types of manipulated resources and executed commands. The command manager plays the role of a transaction manager and is the one responsible for managing the life cycle of a transaction. Transactions in this case span multiple commands and their boundaries need to be defined. While this is feasible, potential complications arise when there are dependencies between commands. If one command is successful and the following one fails, can we blindly roll back all the commands associated with the transaction? In many situations the answer is no. However, in other situations where dependencies aren't a problem, this solution may work out fine.

For additional information on how to leverage the power of Java and XML across distributed transactions, attend my presentation, "Transaction Integrity Across Distributed Java Programs and the J2EE Framework," at JavaCon 2000.

For examples on how to use the Command Pattern and other patterns written in Java see Volume 1 of *Patterns in Java* by Mark Grand (Wiley). And the original patterns book is *Design Patterns: Elements of Reusable Object-Oriented Software* by Gamma et al. (Addison-Wesley).

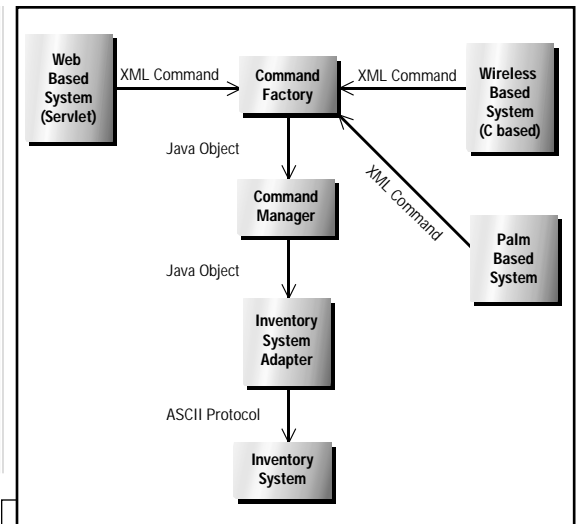


FIGURE 2 Inventory system solutions framework

```
<?xml version='1.0'?>
<Command>
  <Name>Increase</Name>
  <SKU>123-45-6789</SKU>
  <Qty>5</Qty>
  <Warehouse>Dallas</Warehouse>
  <Zipcode>75063</Zipcode>
</Command>
```

FIGURE 3 XML command description

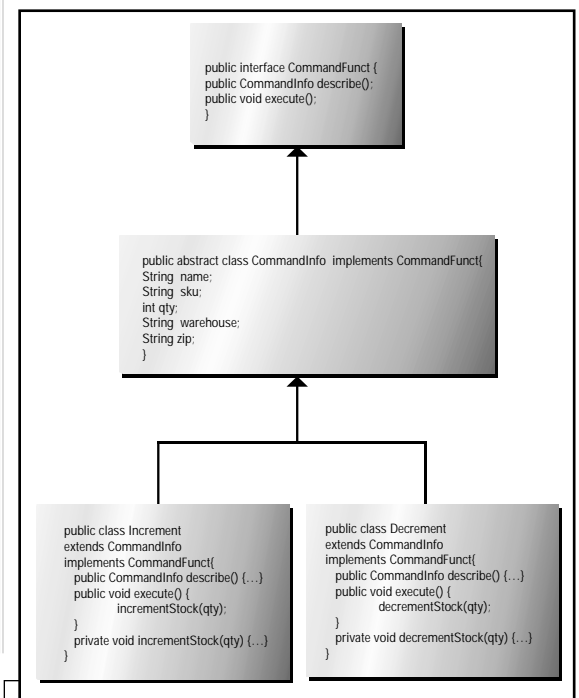


FIGURE 4 Java command class hierarchy

# Wireless Center

# s DevCon Spread



*Is all the effort and disruption worthwhile?*

# XML—Friend or Foe?

**A**t Tradeum Inc. we became early adopters of XML just a few months after the publication of the XML standard in February 1998. Our reasons were compelling: we were creating a dynamic trading engine that implements auctions and true exchanges in a business-to-business environment.

At the heart of the application was a parametric matching engine that matches buyer-and-seller offers based on industry-specific parameters describing the goods to be traded, price and other terms and conditions, and buyer and seller identity. What better data framework for receiving parametric buy-and-sell offers than XML, the framework in which more and more industries would be standardizing data interchanges?

By creating a trading engine that works with any DTD (as opposed to the approach that imposes a DTD such as cXML, which is more suited to catalogued goods), our customers – the market makers and enterprises creating B2B exchanges – can connect directly to buyers and sellers using the standard XML formats in their industry. By embedding the special range and wild card elements that we developed, such as `<range/>` and `<any-element/>`, in the XML documents, buyers and sellers can express flexibility with respect to aspects of the transaction. These are resolved by the system to finally provide a fully defined transaction description in XML.

And yet, despite the strong fit between our objectives and those of the XML initiative, our experiences with XML have been mixed at best. Our frustrations weren't limited to the obvious problems: the lack of tools for a new technology and the surrounding technologies' lack of stability during the last two years. (One example of such frustration leaps to mind – the lack of stability of the XSL transfor-

mation now called XSLT. I once wrote to an XSL tool provider to complain that his tool didn't even recognize the most basic XSL expression `<xsl:process-children./>`, and this had cost us several critical days of development. He wrote back that the working draft had recently changed and this expression was renamed `<xsl:apply-templates/>`!) This article concentrates on inherent issues with the XML standard.

## Semistructured Data

XML defines a syntax for semistructured data centered around the `<xxx>...</xxx>` syntax for elements. *Semistructured* here indicates that specific XML schemas can impose typelike constraints on documents but need not rigorously fix the elements and their types. This offers considerably more flexibility than a programming language object conforming to a class file or a database record conforming to a table definition. Arguably, however, an XML document has considerably less flexibility than a graph of objects or a scheme of database tables.

Coping with the differences in paradigms between object graphs, relational database tables and XML documents will provide employment for software professionals for years to come in a way that only COBOL Y2K-bugs and chasing C/C++ pointers have done up to now. Therefore, XML should be judged on the basis of a cost/benefit analysis as well as on its efficacy. The cost of an entirely new data paradigm is exceedingly high

(witness database vendors struggling to come out with a usable package to store XML documents efficiently in a database without too much human intervention in mapping XML schemas to relational database schemes).

One important innovation (arguably the only innovation) that XML does offer is a formal syntax for defining the semistructured schema to be satisfied by documents for particular applications. This provides flexibility that's useful in allowing variation in data structure from subcategory of goods to subcategory of goods, user to user or system to system. Of course, this flexibility always comes at the expense of making data structures less predictable and processing more complicated.

The use of a formal language to describe the schema allows mechanized parsers to check the validity of an XML document. This schema syntax has hitherto been an awful syntax called DTD, which wasn't even an XML language (supposedly the ultimate language for any semistructured data!), soon to be replaced by a slightly less awful syntax called XML Schema, a well-formed XML (allowing lots of recursive fun with an XML Schema defining the syntax for XML Schema). XML Schema will be widely backed (which probably means that the only name not reserved for a type that someone wants will be `<designed-by-committee/>`).

XML provides a reasonable framework for semistructured data and a standard language (at least two "standard" languages) for defining schemas. The rest of the hype can safely be ignored. For

## AUTHOR BIO

Zvi Schreiber is the founder and CTO of Tradeum Inc., which pioneered the concept of dynamic B2B electronic commerce and adopted XML in 1998. Zvi holds a PhD in theoretical computer science from Imperial College, London. He lives in Jerusalem, Israel.



# ASP WORLD CONFERENCE

[www.aspworldexpo.com](http://www.aspworldexpo.com)

example, XML isn't self-describing in any meaningful way. You can call the element that describes color `<color>` (or, giving away my roots, `<colour>`), but this means nothing to the computers – the document's intended audience – at least, nothing more than having a variable name "color" in your object or a field "color" in your database table definition – no one ever called these self-describing. So XML and the associated schemas are really about syntax, not semantics. The semantics are still described in plain language.

Is XML at least the perfect data language for the specific purpose of the exchange of semistructured data? Unfortunately not, for the simple reason that it wasn't designed for this purpose or, at best, it was derived from languages that weren't designed for this purpose. XML is a subset of SGML and a generalization of HTML, both largely designed for publishing.

Here are some examples of where the publishing heritage of XML gets in the way of the applications for which XML is now touted.

### Attributes or Textual Elements?

We had several debates at Tradeum on whether a range should look like

```
<range min="1" max="10"/>
```

or

```
<range>
  <min> 1 </min>
  <max> 10 </max>
</range>
```

The XML standard doesn't provide any guidance on when to use attributes and when to use textual elements inserted in child elements. The reason is that in the publishing world it was clear: anything that didn't appear on the page (for example, choice of font) went in an attribute and anything that appeared as text on the page (including elements such as the page header) deserved a textual element. When applying XML to abstract data this becomes an annoying matter of taste. Needless to say, we ended up supporting both formats, which probably adds a few microseconds to the processing time of every document passing through our system.

### Ordering of Elements

Another area of confusion is the ordering of elements. Are the element lists `<a/><b/>` and `<b/><a/>` identical? XML specifies that attributes aren't ordered so:

```
<aa b="1" c="2"/>
```

and

```
<aa c="2" b="1"/>
```

are certainly the same, but in general:

```
<a/><b/>
```

and

```
<b/><a/>
```

are different. This can be useful if we want them to list the order in which items should be shipped:

```
<shipping-order>
  <item> ... </item>
  ...
  <item> ... </item>
</shipping-order>
```

However, it's more common to specify attributes by name, and the order is irrelevant:

```
<software-engineer>
  <experience> ... </experience>
  <languages> ... </languages>
  <favorite-pizza> ... </favorite-pizza>
</software-engineer>
```

What difference would it make if the order of the child elements was changed? What counts here is the element name, not its position. However, XML doesn't provide a general way of indicating that data should be viewed as unordered name/value pairs instead of an ordered list. For Tradeum this means that our software has no general way of deciding whether to match a buyer and seller who specify XML documents that are compatible except for the order of elements.

This lack of attention to whether elements have an order isn't surprising, since in publishing everything is naturally ordered by the order in which it's supposed to be read! The concept of unordered elements referenced by name didn't arise in the main applications of SGML and HTML. (On the other hand attributes, which in publishing represent data that wasn't part of the document as noted above, don't have a natural ordering and were declared to be unordered name/value pairs; just what we'd expect from XML elements in most nonpublishing applications!)

This dual and uncontrolled use of XML for ordered and unordered lists makes it awkward to talk formally about an XML document. This forces XSLT and XQL to provide two sets of formal ways to reference an element: using an element name path such as `software-engineer/languages` or using a position such as `[2]`.

(Quiz: How do you mix the two methods and refer to the element in an element list that follows the element named `<start/>`? Answers on postcards please...)

### White Space Issues

XML has other annoyances. The language, which is supposed to be used for communication between computers, is typically formulated with lots of white space to make it indent nicely for human readers. This is a manifestation of the tenth declared design goal of the XML Specification: "Terseness in XML markup is of minimal importance." (One can't help wondering whether this rather negative design goal was added to pad out the number of objectives to 10 – a good example of terseness not being important.) This, of course, necessitates a set of confusing rules determining which white space is part of the text and which is to be interpreted as padding. (At Tradeum every time we thought we understood how XSLT dealt with white space, the standard changed. We've spent half our lives taking `normalize()` functions in and out of XSLT scripts. This said as much about the confusing issue of white space in XML as it did about the instability of the XSLT working draft.)

So there it is. XML offers a data format that's reasonably well suited for human reading (in a data language supposedly designed for computer-to-computer interaction and at the expense of annoying white space issues, etc.), a choice of powerful but somewhat clumsy formal languages for defining semistructured schemas (at the expense of forcing all data into a tree hierarchy and making processing complex) and the ability to handle semistructured data flexibly (but no inherent ability to distinguish between different types of data collection such as ordered lists or name/value pairs). Whether these benefits, such as they are, are worth the disruption of an entirely new data paradigm that's difficult to map to the relational or object models is a matter of opinion.

I'd like to conclude with a positive side to this story. XML has inspired many industry consortia to announce their intention to create standards for exchanging data in their industries (inevitably, on many occasions, several standards per industry). If half the standards announced to the press come to fruition (and a number already have), all the effort and disruption will have been worthwhile and our early adoption of XML vindicated. ☺



# XML DevCon FALL 2000

Register by  
Friday, October 6  
and  
**SAVE \$200**

**Conference:**  
November 12-15, 2000

**Exhibition:** November 13-14, 2000

DoubleTree San Jose Hotel  
San Jose, California

## Plan to Attend the Largest XML Event Coming to Silicon Valley

**...Featuring an All-Star Lineup of Instructors  
and over 100 Participating Companies...**

If there's one XML Conference you attend this fall,  
make sure it's XML DevCon -

*the only XML event to combine the most XML vendors with the most technically advanced, comprehensive technical program. Join 5,000 XML enthusiasts - the industry's most respected technical experts, sought-after gurus and advanced users. Come immerse yourself in four days of the hottest XML techniques and master new skills taught by those who are defining XML's future.*

### Event Highlights

- Back by overwhelming demand, this will be the largest XML event to come to Silicon Valley in 2000.
- Expanded program - 6 tracks, over 100+ sessions to select from.
- Special Preconference Tutorial on Sunday - choose from 11 in-depth sessions.
- Night School Sessions - select from 18 in-depth topics on Sunday, Monday and Tuesday nights.  
Extend your day program or take independently
- Learn from masters. Just take a look at the Faculty List.

### Distinguished Faculty Members Include...

Charles Goldfarb, father of SGML and author of *The XML Handbook*, *XML Web Kit*

Jim Gray, coauthor of *Transaction Processing: Concepts and Techniques* and *The Benchmark Handbook: For Database and Transaction Processing Systems*

Don Chamberlin, codesigned *SQL* and *Quilt* and authored *A Complete Guide to DB2 Universal Database*

Joe Celko, author of *SQL for Smarties*, *Instant SQL*, *Data and Databases*, and *Joe Celko's SQL Puzzles & Answers*

Kurt Cagle, author of *XML Developer's Handbook*

Michael Floyd, author of *Building Web Sites with XML*

Elliott Rusty Harold, author of the *XML Bible*, *XML: Extensible Markup Language*

Ken Holman, author of *Practical Transformation Using XSLT and XPath*

Bruce Peat, coauthor of *Professional XML*

Simon St. Laurent, author of *XML: A Primer*, *XML Elements of Style*, *Building XML Applications*, *Cookies*, and *Sharing Bandwidth*

Tommie Usdin, editor of *Markup Languages: Theory & Practice*

Paul Brown, coauthor of *Object-Relational Databases: The Next Great Wave*

### Explore Today's Most Relevant Topics Including...

- \* Bleeding Edge of XML
- \* Building Web Sites with XML
- \* XML for Programmers
- \* Introduction to XML
- \* Practical Transformations Using XSLT and XPath
- \* Integrating XML, Java and CORBA as a Distribution Mechanism
- \* XML vs EDI
- \* Enterprise Data Access and Content Syndication
- \* Using XML to Facilitate Data Flow
- \* BizTalk Development
- \* Tree-Based XML Processing with Perl
- \* Generating XML from Form Data
- \* Application Integration Using XML
- \* XML and JavaServer Pages
- \* XML Meets Middleware
- \* XOC and B2B Collaboration
- \* DOM Level 2
- \* Where's the Data? Trading Between Relations and XML
- \* OASIS Standards Activities
- \* XML APIs in the Java Software Platform
- \* Financial Industry B2B Messaging Standards
- \* Lycos Multitier XML Architecture
- \* SMIL - You are Seeing Candid XML
- \* Understanding Quilt
- \* EAI Using XML and XSL
- \* Content Generation and Distribution Using XML and Jini
- \* Integration Services, Wireless and Databases
- \* Using WebDAV
- \* Programming for Browser Interoperability
- \* XML and Open Source Medical Records
- \* Automatically Creating Servlets from XML
- \* Java, B2B, Wireless Drive eBiz
- \* Understanding SOAP
- \* Querying XML Documents
- \* Standard XML Templates for Healthcare
- \* XML and Scripting Languages
- \* VoiceXML
- \* XML and Stylesheets
- \* Structuring Documents for Databases
- \* Topic Maps, Portals and Info Management
- \* Using SAX and Java to Explore XML
- \* Building Wireless Applications Using JSP
- \* Advanced SOAP Programming
- \* Interface Development with XUL
- \* Mapping DTDs to Databases
- \* Networked Marketplaces
- \* Java-Based Xbeans
- \* 21st Century XML Servers
- \* XML-Bots
- \* XHTML: XML for Client-Side Authors
- \* Serializing Query Results in XML
- \* B2B eXchanges
- \* ebXML & JAXM-eCommerce and Java
- \* XML Support in TSpaces
- \* XML Views in SQL Server 2000
- \* Hows and Whys of XML Specifications
- \* e-Business Initiatives and ebXML Infrastructure
- \* 4th Generation Application Development with Java
- \* XML-centric Architectures for the Web
- \* Applying XML to Web Collection Indices
- \* Schemas: Rules for XML Documents
- \* Real-Time eBusiness for Competitive Advantage
- \* Sequential XML Processing and XML Standards
- \* Introduction to RDF
- \* Repurpose Your Data: Introduction to XSL
- \* Using Statistics for Structure Mapping



### 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.

*Every delegate receives a free one-year subscription to XML-Journal and a one-year FREE subscription to Java Developer's Journal - a \$99 value!*

**WWW.XMLDEVCON2000.COM or INFO@CAMELOT-COM.COM**

Presented by: **SYS-CON MEDIA**  
**XML JOURNAL**

Premier  
Sponsor:



Corporate  
Sponsors:



Cosponsor:



Produced by:





*Using XML to define data hierarchically  
can produce highly redundant elements*

# Eliminating Redundancy in XML Using ID/IDREF

**X**ML can be thought of as the “universal serialization of data.” It provides a flexible, open approach for modeling data and sharing messages among business partners (or systems) in a consistent manner. XML provides the ideal solution to messaging in a B2B e-commerce infrastructure since it enables a loosely coupled design that can significantly lower a partner's barrier to entry.

## AUTHOR BIOS

John Evdemon is the chief architect for XMLSolutions Corporation ([www.xmls.com](http://www.xmls.com)), a product and services firm headquartered in McLean, Virginia. He has been designing and deploying enterprise systems on a variety of platforms for over 11 years.

JP Morgenthal is executive vice president of development for XMLSolutions. He recently authored Enterprise Application Integration with XML and Java for Prentice-Hall.

While most users of XML utilize its hierarchical nature to define data, this article discusses possible approaches for eliminating redundant data within XML messages by employing the features of ID and IDREF to define a more relational approach to defining the data.

## Reporting Data Is Redundant

Developing a common vocabulary for a single enterprise or a large B2B marketplace can be a daunting task. With XML the challenge of modeling commonly utilized data models in an open manner can be resolved. It's based on a formal W3C Recommendation and has extensive support from both users and IT vendors, thereby providing a stable, future-proof foundation for business applications. The technical and business benefits of utilizing an open, standards-based XML strategy are immense. Whenever possible, an enterprise's existing data models and semantics should be utilized to construct the markup language.

There are, however, several issues that must be addressed prior to effectively representing an organization's

data models in XML. Common reporting and aggregate data sets frequently contain repetitive data such as name and address. These redundancies are usually carried over into the resulting markup. This approach leads to repetitive data structures – the result of which is poorly designed markup that doesn't make use of reusable data models. XML's ID and IDREF tokenized attributes enable designers to leverage relational modeling concepts and avoid creating redundant data structures.

ID is a specially defined attribute that uniquely represents an instantiation of

an element within an XML message. The value of an ID attribute is similar to a primary key value in a relational database table in that no two ID attributes (for a given element) can have the same value. IDREF is another special XML attribute that references a previously defined ID value. IDREFS is similar to IDREF, but can point to several previously defined ID values, each separated by a space. Listing 1 displays a simple example of how ID, IDREF and IDREFS are typically used in an XML document.

As illustrated in Listing 1, ID, IDREF and IDREFS can be used to establish relationships within the data. They can also be used to represent data structures without utilizing nested elements. This approach enables developers to model data that lacks definite structure (such as

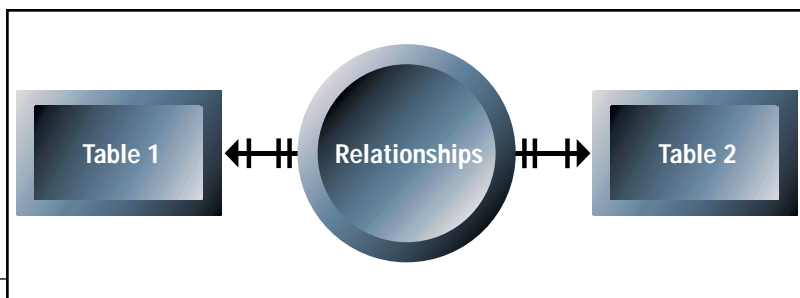


FIGURE 1 Applying relational database design concepts eliminate redundant data structures in your markup



a result set from a user-defined query).

Based on these concepts, a set of redundant elements can be effectively modeled using ID and IDREF attributes. A sample document with redundant elements appears in Listing 2. As this listing clearly illustrates, the Key, Firstname and Lastname elements appear in several locations throughout the document.

Listing 3 illustrates how this data can be modeled in a more effective manner using ID, IDREF and IDREFS.

Note that the code in Listing 3 establishes a data structure (Keys) that can be reused elsewhere in the document. This design reflects the relational approach to defining and utilizing translation tables to avoid many-to-many relationships (as seen in Figure 1).

## Processing XML Using Xpath and ID/IDREF

Today's XML parsers don't make use of the ID/IDREF functionality as defined in this article. However, the W3C XPath specification ([www.w3.org/TR/1999/REC-xpath-19991116](http://www.w3.org/TR/1999/REC-xpath-19991116)) provides facilities for identifying and selecting nodes based on their IDs. Additionally, IDREFs can automatically be retrieved from identified nodes and supplied directly as input into the ID function.

The XPath specification defines a set of core functions that must be available

# The XPath specification

to all implementations of XPath processors. One of these functions operates over an XML document to provide a node-set result based on the ID function. The ID function takes a single string parameter identifying the node that defines the unique ID. Using the example in Listing 1, we could use the following XPath statement to retrieve the supervisor for a particular employee:

`id("456")` returns the SUPERVISOR node for Debbie Hamel

The input to the ID function can also be a white space-separated list of strings that represent multiple IDs and the resulting node set would include all nodes that have IDs matching those defined in the list.

## Conclusion

As you can see by the differences in Listings 2 and 3, using XML to define data hierarchically can produce highly redundant elements. By using the method described in this article, it's possible to mix hierarchical and relational techniques within the same document, which results in a more usable and reusable XML document. In addition, this document will contain clearly recognizable elements that can be employed to describe information relative to data both within and external to the XML document. ☛

JOHN.EVDEMON@XMLS.COM

J.P.MORGENTHAU@XMLS.COM

### LISTING 1

```
<?xml version="1.0"?>
<!DOCTYPE SAMPLE [
<!ELEMENT SAMPLE (SUPERVISOR+, EMPLOYEE+)>
<!ELEMENT SUPERVISOR (NAME)>
<!ATTLIST SUPERVISOR ID ID #REQUIRED>
<!ELEMENT EMPLOYEE (NAME)>
<!ELEMENT NAME (#PCDATA)>
<!ATTLIST EMPLOYEE SUPERVISOR IDREF #REQUIRED>
]>
<SAMPLE>
  <SUPERVISOR ID="123">
    <NAME>Jeff Ricker</NAME>
  </SUPERVISOR>

  <SUPERVISOR ID="456">
    <NAME>Debbie Hamel</NAME>
  </SUPERVISOR>

  <EMPLOYEE SUPERVISOR="123">
    <NAME>Dan Cocos</NAME>
  </EMPLOYEE>

  <EMPLOYEE SUPERVISOR="456">
    <NAME>Yoshi Russel</NAME>
  </EMPLOYEE>
</SAMPLE>
```

### LISTING 2

```
<Database>
<View_1>
<Key>1232231</Key>
<Firstname>Bruce</Firstname>
<Lastname>Wellington</Lastname>
<Address>161 West Park Drive</Address>
```

```
<City>South Park</City>
<State>CO</State>
<Zip>30303</Zip>
</View_1>
<View_2>
<Key>1232231</Key>
<Firstname>Bruce</Firstname>
<Lastname>Wellington</Lastname>
<PurchaseOrderNumber>122343</PurchaseOrderNumber>
</View_2>
</Database>
```

### LISTING 3

```
<Database>
  <Keys>
    <Key ID="1232231">
      <Firstname>Bruce</Firstname>
      <Lastname>Wellington</Lastname>
    </Key>
  </Keys>
  <DataView>
    <Address IDREF="1232231">
      <Street>161 West Park Drive</Street>
      <City>South Park</City>
      <State>CO</State>
      <Zip>30303</Zip>
    </Address>
    <PurchaseOrderNumber IDREF="1232231">
      122343
    </PurchaseOrderNumber>
  </DataView>
</Database>
```





*An essential technology with a mission*

# XML: True Collaboration

Lo and behold! XML has found a home and is beginning to make an impact on the enterprise. XML used to be considered fabulously interesting (like a degree in philosophy); now it's considered an essential item in everyone's bag of tools (like a wrench). As this technology hits the next level of maturity, it's time to focus on its unique business benefits.

## Where Are We Now?

XML is firmly in the early adopter phase. That used to be the first phase, but clearly XML has existed for a long time in an *academic* phase. We've known how to spell "XML" for over three years and worshiped it for over two – but what was really going on?

XML, as it's used today, didn't evolve smoothly. It originated as a simplified version of SGML and was poorly understood during much of its infancy. Initial expectations were that it would be a publishing language or HTML on steroids, and its chief benefit would be better searching on the Web. Due to significant exploration and academic-style research to find the biggest bang for the buck, XML didn't take that road.

Yes, XML can be thought of as the oat bran for whatever ails you, but it's launching its strongest foothold in business-to-business collaboration. XML is no longer thought of in terms of its technological features but its business benefits. They completely leverage the unique and revolutionary features that account for the posthype success.

In the academic phase XML was used primarily by intellectual geeks, daring independent software vendors and boutique system integrators who had the freedom to experiment without the risk

of significant long-term consequences. Its exposure to corporate managers was limited to a skeptical reading that provided fodder for cocktail parties. Now that XML has graduated from just features to real benefits, the early adopters within the enterprise are starting to put XML to the test.

## True Collaboration

Businesses and diverse departments have worked together for years – often electronically. So how is true collaboration (or collaborative commerce, c-commerce – choose your favorite buzz word) different?

Compare how you interact with the guys down the hall versus co-workers in remote offices. Remember the old adage, "There's nothing like being there"? When you have the luxury of proximity, you tend to develop and depend on an ad hoc relationship. You share unpredictable things at unpredictable times. Remote relationships tend to be more formal, and electronic relationships are more formal still. You share predetermined business documents, contact each other when there's an official reason, and engage in a predictable and often rigid protocol.

Look in your file cabinet and see what goes into a typical manila folder – all

kinds of odd *stuff*. When working with others in a paper-based world, you have the luxury of ad hoc contributions beyond just business documents, such as photographs, cocktail napkin diagrams and additional notes in the margins. It's sharing these bits and shreds of unusual, informal and unpredictable content that enables the richest form of collaboration.

The difference between the electronic interaction that's been possible in the past and the collaboration envisioned for the future is the ability to interact and share information with as much freedom and ease as you can with office mates.

As industries become increasingly competitive, companies are looking to cut costs and improve services like never before. Every ounce of efficiency has to be squeezed out and every option delivered. This often requires resources beyond the enterprise, and enabling the extraprise to work as productively as entities within the four walls of your company is what true collaboration is all about.

This is the business benefit that XML uniquely enables.

## XML: Enabling Collaborative Commerce

By now most of us are familiar with XML's basic features. Yes, it's simple, standard and platform independent, but those aren't revolutionary features. They're requirements of any business-to-business data format. These features alone can't make collaborative commerce possible. If that were so, EDI

## AUTHOR BIO

Coco Jaenicke was, until recently, the XML evangelist and director of product marketing for eXcelon Corporation. She played a key role in the successful development and introduction of eXcelon, the industry's first application development environment for building and deploying e-business applications. She recently joined SilverStream Software.



would have done the job. Arguably, EDI isn't simple or Web-based, but evolution could have taken care of those deficiencies. No, something else enables XML to revolutionize interaction.

XML possesses two killer features. One, it's flexible enough to handle any information no matter how unwieldy, oddly structured or bizarre. Two, it's extensibility enables it to handle the unpredictable as well as on-the-fly, ad hoc additions. (Refer to my earlier column entitled "XML: It's the 'X' that Matters" in the premier issue of *XML-J* [Vol. 1, issue 1] for an in-depth diatribe on the undeniable virtues of extensibility.) These features are truly new to the enterprise.

This is where XML makes the jump from a neat idea with an academic following to an essential technology with a mission. If the essence of collaboration is being able to share rich content (an elegant way of saying "Any kind of crap you can imagine") and to interact unpredictably (freedom from schema!), then it's clear how XML fits in. It facilitates the electronic equivalent of throwing a sketch over a cubical wall or scribbling a note of wisdom on a folder.

Consider a few business-to-business examples. If you share an insurance form with another business, you may also want to include a photograph or medical data. An RFP may depend on CAD diagrams or architectural plans. A business document could be returned with an additional section inserted in the middle. Another partner may need to read that document too – but isn't privy to the new section. In general, XML lets you think about collaboration, not as sharing rigid, predetermined business documents, but as sharing a dynamic manila folder filled with rich content.

Because of XML, that folder is flexible enough to contain any type of information, and extensible enough to easily manage all value-add opportunities. This is that next level of collaboration.

## Making It Happen

Having XML somewhere in your building is not enough to ensure that your infrastructure will deliver on these benefits. XML needs to be in certain places at certain times – having an adapter on your firewall that translates everything into XML at the last minute isn't enough. Keep asking yourself about your strategy: Can you share what you want, when you want, how you want?

What's required is true management of XML in the middle tier or – to stretch the metaphor – a filing cabinet for all the manila folders. It's this filing cabinet – or repository – that makes it possible to add, link and personalize information. This is what allows you to go beyond sharing just the predefined invoice, RFP or PO and collaborate electronically.

Collaboration is what XML brings to the business community. It has found an application – the first of many – that puts its unique features to the test. ☎



# 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



September 24-27, 2000

Santa Clara Convention Center  
Santa Clara, CA

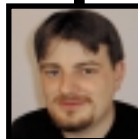
Sean Rhody Editor-in-Chief, *JDJ*

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



Alan Williamson Straight Talking Columnist, *JDJ*

Alan, the Straight Talking columnist of *JDJ*



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

Ajit is the founding editor of *XML-Journal*



Jason Westra EJB Home Columnist, *JDJ*

Jason is the



## ADVERTISERS INDEX

ADVERTISER	URL	PH	PG
AIR2WEB	WWW.AIR2WEB.COM		17
CAPE CLEAR	WWW.CAPECLEAR.COM	353.1.241.9900	15
CAREER OPPORTUNITIES			81
ASP WORLD CONFERENCE	WWW.ASPWORLDEXPO.COM	877.696.0459	47
GEEK CRUISES	WWW.GEEKCRUISES.COM	650.327.3692	33
IBM	WWW.IBM.COM/DEVELOPERWORKS	800.772.2227	84
ICON INFORMATION SYSTEMS	WWW.XMLSPY.COM		4
INFOSHARK	WWW.INFOHSHARK.COM	888.DATASHARK	9
IXIASOFT	WWW.IXIASOFT.COM		31
JAVACON 2000	WWW.JAVACON2000.COM	212.251.0006	54,55
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	63,77
NETDIVE	WWW.NETDIVE.ORG		23
PROGRESS	WWW.SONICMQ.COM/AD11.HTM	800.989.3773 EXT. 450	3
SILVERSTREAM	WWW.SILVERSTREAM.COM		13
SOFTQUAD SOFTWARE	WWW.SOFTQUAD.COM/PRODUCTS/XDMETAL/EVAL/	800.387.2777	2
SYS-CON MEDIA, INC.	WWW.SYS-CON.COM		80
THE SHORT LIST	WWW.THESHORTLIST.COM		21
XML BOOT CAMP	WWW.SDEXPO.COM/BOOTCAMP	415.905.2702	37
XML DEVCON FALL 2000	WWW.XMLDEVCON2000.COM	212.251.0006	49
XML GLOBAL TECHNOLOGIES	WWW.GOXML.COM	206.352.8334	83

# JAVACO

[www.javaco.com](http://www.javaco.com)

# ON 2000

on2000.com



*Using XML to develop a single Java Servlet to serve all delivery channels*

# The New Requirements of Internet Business Architecture

**T**he drive has begun for a new generation of Internet commerce architectures. The stimulus is coming from organizations building customer-oriented e-tail sites and business-to-business applications. New architectures are needed to accommodate both sets of requirements. Java's move into enterprise software development is also a major step toward the development of these new applications. With the XML revolution, major vendors and organizations have agreed on XML as the standard for information exchange. This standardization also requires a change in the way we develop Java-based Web applications. Java Servlets, the part of enterprise Java responsible for interacting with HTTP requests, should be able to communicate with any type of delivery channel such as a customer browser, a PDA/mobile phone or another Web server. With XML as an intermediate format, you can take data from any source and deliver it to any device.

Suppose you have a Web application that provides an online pizza inquiry and order service. You want to deliver this service to a mobile phone, PDA and/or Web browser. Another supplier – a business partner – may be required (order distribution to multiple suppliers) to make specialized pizzas. Since each client device has specific formatting requirements, the same data can't be sent to everyone. The mobile can understand only WML; the browser can render only HTML; and the business partner's Web server requires plain XML data. If you're developing separate servlet (Web application) versions for each delivery channel or using DCOM/CORBA to integrate the different supplier networks, then your architecture needs a major change – XML over the Internet.

XML has gained widespread popularity as a way to represent data in a portable, vendor-neutral, readable format and to exchange data between different systems. It's effectively turning the Internet into a middleware of Web applications in which XML is the spoken language.

This article assumes that readers have a sound understanding of XML/XSL and Java Servlet concepts. I provide a small pizza order and inquiry example to demonstrate how we can use XML to develop Web applications that serve any client device or delivery channel. The code for this article can be downloaded from the **XML-J** Web site. It requires a basic understanding of the Servlets, XML4J and XSL code level.

## XML Processing in Java

Several XML parsers/converters are available. I chose the IBM XML Parser for Java and the Lotus XSL Processor to format information. (Both are available from the IBM alphaWorks Web site, [www.alphaworks.ibm.com](http://www.alphaworks.ibm.com).)

## Traditional Web Application Development

A restaurant owner wants to provide an online pizza inquiry and order service over the Internet (Web browser). Since you know how to work with servlets and Java, you design a domain object model

and write servlets with validation- or order-processing logic. In this case I've come up with Pizza, PizzaType, Order and Customer domain objects. You can have any other classes in your case as required. I've come up with one inquiry – a receiver servlet that generates the HTML dynamically, contains information about different pizzas and can be read from a database. For the sake of simplicity I've hard-coded the two pizza objects (see `getAllPizzas ()` method on the **XML-J** Web site). The `getAllPizzas ()` method returns an array of Pizza objects. Each instance of these objects forms a row in the HTML table and the Pizza types for that object are displayed.

The output of the Pizza inquiry can also be used to submit Pizza orders to the server. I haven't generated any HTML to submit orders; I leave it as an exercise for the reader. I assume that whenever an HTTP request is made to the `AddPizzaToBasket` Servlet, it contains three parameters (Pizza unique name, size and quantity). An order object is constructed against these parameters and added to the session vector that's maintained as a session variable. The request to the `CheckoutServlet` should have the customer name and credit card number. The customer object is constructed against these parameters and a `processOrders ()` method is called. This method is left empty so you can do anything you want

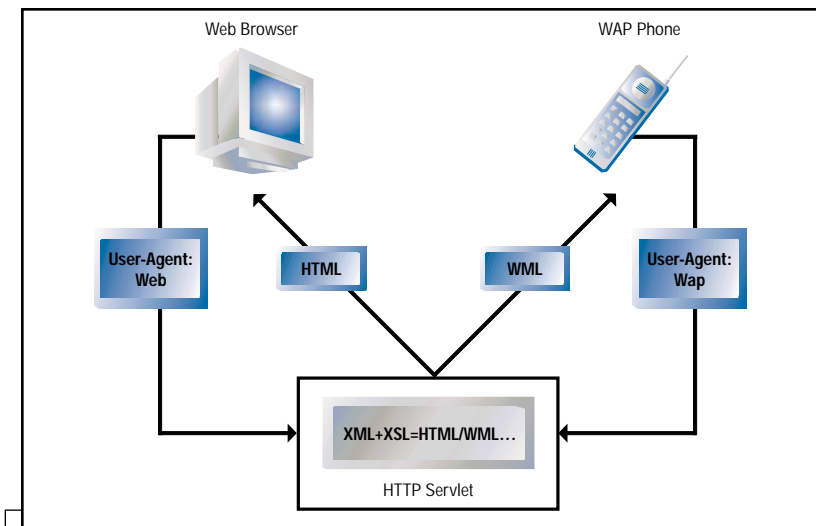


FIGURE 1 XSL converts XML into other formats.

in it. The code for these servlets can be found in Listing 1.

## New Business Requirements from Business People

Suppose you're an architect. The restaurant management talks to you about WAP technology, which would enable pizza inquiries and orders on mobile phones. They want this functionality available as early as possible in order to remain competitive. You have a problem: the WAP-enabled mobile phones can understand only WML and all your HTTP servlets are responding with HTML.

### THE UNSTRUCTURED APPROACH

You develop a separate servlet for the new WAP delivery channel. All problems are solved – you copy and paste all the business processing logic from the HTML-generating servlet and replace the HTML syntax with WML syntax.

Another approach: the HTTP header contains information about the User Agent, which you can retrieve from the HTTP header in the servlet code. Make two separate functions: one for Web channel response generation and the other for the WAP channel response channel. Call them from the `doGet()` function that's based on the User-Agent value in the HTTP header (found in Listing 2).

### XML-BASED STRUCTURED APPROACH

Why have I called the above approach unstructured? A good enterprise application design can absorb new business changes with minimal effect and scale to  $n$  tiers as the business expands.

As an architect, you'll foresee that the business people will come to you again with new requirements. Today they

require a WAP-enabled application, tomorrow – B2B. Look at the unstructured solution again; it'll create problems in the B2B-based architecture extensions since the data is in the form of Java objects. These objects require RMI to travel over the wire. The other end also needs RMI infrastructure, and the solution won't work behind the firewall. The business needs a global data carrier – HTTP and language- and platform-independent data structures – such as XML.

Imagine the scalability of your structured solution. Your Web server will deliver plain XML to any other B2B partner over HTTP (the Internet). This solution works for  $n$  tiers of business-logic processing in which each tier is from a different party and resides behind the firewall. XML data is independent of any client device so the delivered information can be formatted/parsed to be used in any application (HTML, WML, PDAs, ATMs, FAT/custom applications). The business-logic-generated XML data is used by all delivery channels. Only one copy of business logic is used by all types of clients. Any change in the business rules is automatically applied to all channels.

### XML IN INQUIRY APPLICATIONS

The restaurant owner wants to serve mobile phone clients. The information for both distribution channels (Web and WAP) is the same, but each requires different formatted data. Here XSL comes into action. It has the ability to convert XML data into any other suitable format as required by the delivery channel (see Figure 1). The business logic to extract information from the data source remains at one place, the same servlet can serve multiple clients, and information for each of them can be formatted using XSL (found

in Listing 3). All the WML tag code is tested with Nokia WAP toolkit version 1.2.

### XML IN ORDER PROCESSING

You've just seen XML usage in the information delivery, but what about information processing? The order object is constructed whenever the request is submitted to the `AddPizzaToBasket` servlet. The vector of the order is maintained in the session. The hit to the `CheckOutServlet` retrieves the vector from the session and processes the orders. However, we want a scalable design that distributes an order to multiple suppliers who process it in an optimal manner. These suppliers may reside behind firewalls and possess different IT infrastructures. XML over HTTP can play a significant role here. The order's DTD is distributed to all suppliers and XML is sent as the parameter in the HTTP request. The response also contains the result in XML, as this will be a B2B message or transaction.

While I was writing this article, a friend asked, "Why are you using XML to send the data to the other Web server over HTTP? Why not use the simple form field/value parameters?" I agree. This is true; a simple field/value pair can do the same task but this approach is useful for simple form data submission. What would happen if the orders have complex structures made up of multiple interrelated parameters? Where will the structure information about the data be maintained, or how will you relate one parameter to another? If you use the simple field/value parameters, the HTTP request handler code at the other partner's server should have the necessary information about the interparameter relationship. Any change in the parameters will require code changes in all the receiving Web servers. This would be a bad design from a maintenance point of view. In the case of XML the change requires only an update in the DTDs, which will be distributed to all suppliers, but the code for input validation or data extraction remains as is. Listing 4 shows the code.

It's a simple process to send XML data as form data over HTTP to the partner's Web server. The response is also received as XML data. In addition, I've developed order and order acknowledgment DTDs that validate the structure of submitted or received XML data. To keep the examples simple, I've hard-coded the DTD reference paths in the XML data or document. You may want it to be separate but I'm a bit lazy! ☺

### AUTHOR BIO

Syed Fareed Ahmad works as a Java and XML programmer and designer in Pakistan. He focuses on server-side Java technologies to develop Web-based business applications.

SYED\_FAREED@YAHOO.COM



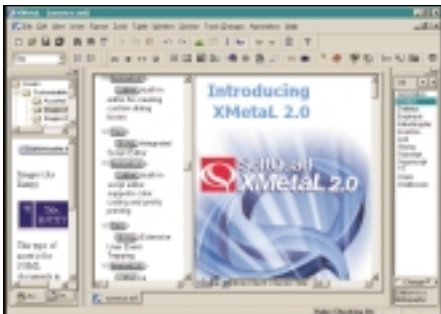
## The NeXML Generation: XML Software Grows Up

A new generation of XML software is arriving on desktops, sporting sophisticated interfaces and advanced integration with database and content management tools. SoftQuad is shipping release 2.0 of XMetaL, its award-winning XML/SGML editor. Extensibility has released Turbo XML, a suite of XML solutions including XML Authority, XML Instance and XML Console. And a major upgrade to XML Spy, the XML IDE, has arrived from Icon Information-Systems.

### XMETA L 2.0

SoftQuad's customizable editor is making steady inroads with content creators. Its familiar word processing-style interface hides a rich array of authoring aids in a highly programmable package suitable for all levels of users.

Version 2.0 adds a document outline view that can be opened by dragging a small vertical bar from the corner of the document window, making document navigation easy. Files identified by a URL can now be opened by double-clicking. Customizations of the menus and toolbars can be inherited by all documents, and the User Guide and Programmer's Guide are now provided in HTMLHelp format. You can provide separate custom help files and context-sensitive help buttons.



Version 2.0 sports a built-in macro editor and a form editor for creating dialog boxes invoked from XMetaL. The integrated script editor supports color coding and context-sensitive editing. Scripts can be triggered by a broad range of user events such as mouseovers and right-clicking. ActiveX controls, such as image viewers, and XMetaL's Calendar Builder can be displayed directly in the document window. Custom tabs, which can host embedded ActiveX controls, can be added to the Resource Manager.

Windows Shell Namespace Extensions are now supported by the Resource Manager, and

XMetaL will now retrieve DTDs specified using URLs. XMetaL 2.0 supports most of the W3C's CSS1 and some aspects of CSS2, including auto-numbering and CSS tables. Users can quickly edit CALS and HTML inline tables with the Table toolbar.

Well-formed XML documents can be opened and edited without a DTD, and Unicode support has been improved. Files that use the ISO Latin-1 character set and are opened in UTF-8 or UTF-16 format can be saved in the same format. XMetaL 2.0 uses Microsoft's MSXML DLL for XSLT support. DOM (Level 1) support is much enhanced, and OASIS catalogs can be used to resolve external identifiers in an XML/SGML file.

XMetaL is having an impact on Capitol Hill. *The Congressional Quarterly* ([www.CQ.com](http://www.CQ.com)) is to Washington insiders as *The Hollywood Reporter* is to aspiring filmstars. It provides a nonpartisan game card listing all the Congressional committees, their members, which legislation got green-lighted and by whom, the budget and the anticipated box office at the next federal election.

Thomas Technology Solutions, Inc., has implemented a new Oracle database for CQ that stores incoming content as XML documents. CQ staff use XMetaL to edit and help automate the content routing and approval process. XML markup means CQ's content can be shared easily between its many publications and Web sites.

USAToday.com is also using XMetaL to manage its online news Web site. Taking advantage of XMetaL's programmability, workflow and multiple approvals have been automated, transforming their static HTML site into a dynamic XML environment.

SoftQuad recently announced a partnership with Documentum, Inc., to integrate XMetaL 2.0 with Documentum's 4i eBusiness Edition (see OASIS article on next page). Documentum specializes in "Internet-scale" content management solutions for e-business, but needed a powerful front-end authoring tool to keep the entire process in XML.

### TURBOXML

It's about time a vendor brought back the "turbo" designator, which used to be so popular in software product names! Extensibility, Inc. ([www.extensibility.com](http://www.extensibility.com)), has introduced TurboXML, a suite of XML solutions including XML Authority, XML Instance and XML Console. Extensibility's specialty is in XML schema development, conversion and management –

enabling enterprises to redefine themselves by transforming infrastructures to gain competitive advantage.

XML Authority, Extensibility's flagship product, validates and converts schemas between all major and emerging schema standards, allowing e-commerce applications, for example, to transact with XML implementations using different schema. Authority can import data from documents structured by DTDs, XDR, BizTalk or custom input from existing applications, and output schema information for current and proposed standards including SOX2, XSDL, DCD, DDML and RELAX. (A trial version of XML Authority is included with XMetaL 2.0.) Unlike many of today's XML products, XML Authority supports UNIX and Macintosh operating systems as well as Windows.

XML Instance is one of the first schema-driven data editors, intended for structured data-oriented documents such as invoices. XML Console is an XML development environment for the desktop of an individual or a workgroup. XML Console includes SchemaDOC, a visual data dictionary editor for schema components.

### 9 1/2 WEEKS IN THE LIFE OF A SPY

Nine and a half weeks after the release of XML Spy 3.0, over 23,000 users have downloaded "the first true IDE for XML," says Alexander Falk, president of Icon Information-Systems ([www.icon-is.com](http://www.icon-is.com)). Over 7,000 licenses to Spy had been sold by the end of July 2000, an indication of the growing interest in sophisticated, second-generation XML tools.

XML Spy is a validating XML editor for Windows, with four advanced document views: a Grid View provides for structured editing, a Database/Table View shows a table of repeated elements, a Text View displays syntax-colored source code, and a Browser View uses Internet Explorer 5 to render XML documents inside XML Spy.


Spy's editor sports a nifty auto-complete feature that presents appropriate coding options as text is entered, based on the DTD or schema in use. Major schema dialects such as DCD, XML-Data, XDR, BizTalk and the April 7 W3C XML schema draft are supported. Spy can import and export text files, Word documents, Access files and ADO/ODBC-compliant databases such as Oracle and SQL Server.



# XML NEWS

Falk believes that, unlike XMetaL, XML Spy is primarily a developer tool. "XMetaL is really focused on content editing for people who don't want to dive too deeply into XML," he told **XML-Journal**. "It competes with FrameMaker or WordPerfect, as it offers technical writers or content creators a WYSIWYG XML editor."

"The reason we call XML Spy an IDE is because it integrates the three major aspects of XML - XML editing and validation, Schema/DTD editing and validation, and XSL editing and transformation. You could use XMetaL plus Stylus plus XML Authority, or you can use XML Spy and have everything in one integrated tool. You can jump to the definition of any XML element or attribute. With one click on the toolbar you are transported right into the DTD or schema."

XML Spy works with any external XSLT processor. Icon plans to release a COM-based API to let developers integrate XML Spy with other products. A single-user Lite version is only \$39, and a 30-day evaluation version can be downloaded for free ([www.xmlspy.com](http://www.xmlspy.com)). 

## i4i Enhances S4/TEXT "Tagless" XML to Speed Patent Process

Like most government agencies around the world, the U.S. Patent and Trademark Office (USPTO) is eager to conduct e-business. Especially since its rapidly increasing workload can be credited to a boom in e-patents. Currently, patent applications can take four years or more to be processed, and much of that time and effort involves looking for paper, reproducing paper and shipping paper.

"Right now we have four and a half acres of office space devoted to paper," says Dennis Shaw, chief information officer of the USPTO. Facing a 75% increase in workload over five years and continuous budget cuts, his goal is to conduct 80% of the office's transactions electronically by 2003.

In almost every scenario you can think of, Shaw has a big problem. Should he scan that mountain of paper and hire a legion of office workers to enter data, check and correct errors?

The solution he chose, the latest release of S4/TEXT, will be introduced at XML World 2000 this month in Boston by i4i (Infrastructures for Information, Inc., [www.i4i.com](http://www.i4i.com)). i4i's technology accomplishes what at first glance appears impossible - applying SGML or XML markup without changing the original file, and presenting the user with almost no hint that XML markup is occurring - "tagless" XML.

Earlier reports have described i4i's product as simply a Microsoft Word plug-in, which is about as useful as describing the space shuttle as a large, white delivery vehicle. i4i uses a patented technique called DataPipes to create a parallel stream of metadata describing the original file but leaving it intact.

This means that Mr. Shaw's mountain of paper, much of which already exists as office automation files of some kind or other, can remain just as it is. i4i's markup process has no effect on the original file, but provides all the power and functionality of a standard SGML or XML implementation.



## "DNS" FOR XML: OASIS LAUNCHES XML REGISTRY

These are interesting times, especially for critics of the next wave of computing. Faced with open technologies not controlled by single vendors, a spate of computer columns was recently published on the "inevitable fragmentation" of standards such as XML and Linux. Clearly not understanding that "X" stands for "eXtensible," XML's critics have pointed to the proliferation of industry-specific XML standards as evidence that the new markup language might fail to achieve ubiquity ([www.zdnet.com/pcweek/stories/news/0,4153,2551691,00.html](http://www.zdnet.com/pcweek/stories/news/0,4153,2551691,00.html)).

**"Sea Change: A profound transformation"**  
—Shakespeare

**"The Times They Are A-Changin'"**  
—Bob Dylan

**"It's outta control!"**  
—Regis


What some see as a problem, others praise as a virtue. Like the Internet, XML and Linux were designed to grow and adapt. What is new is the rise of cooperative mechanisms to help manage and promote these adaptations.

OASIS, the nonprofit XML international consortium founded in 1993 to advance structured information standards, is meeting the challenge head-on with its XML.ORG Open Registry and Repository for XML Specifications and Vocabularies (<http://xml.org>). Developers and standards bodies are invited to publicly submit, publish and exchange XML schemas, vocabularies and related documents.

Developed by Documentum and Sun Microsystems using iPlanet and Oracle 8i, Documentum's 4i eBusiness Edition provides content management for the Registry. Hardware servers were provided by Sun Microsystems and staff were contributed by IBM and DataChannel.

Jon Bosak, one of the earliest figures in the history of XML, admits the rapid development of new schemas, vocabularies, namespaces and stylesheets for XML has raised concerns. "By providing the community with a noncommercial resource for accessing these XML specifications, the XML.ORG Registry serves as an essential piece of the XML infrastructure," he said. Someone in a particular industry might see in the Registry several useful XML schemas and be inspired to collaborate in their evolution.

Now open for contributions, the Registry can be searched. You can download XML specifications such as vocabularies, DTDs, schemas and namespaces for a wide variety of applications. There's no charge to register your work, and contributors retain all rights to their work and control over its use.

The new Registry is designed to serve as a model for an extensive network of XML registries and repositories on the Internet, much like the Domain Name System (see [www.dns.net/dnsrd/](http://www.dns.net/dnsrd/)), a distributed Internet directory service managed by a worldwide network of servers. Specifications for this network are being developed by the OASIS Registry and Repository Technical Committee ([www.oasis-open.org/html/rpublic.htm](http://www.oasis-open.org/html/rpublic.htm)), led by the National Institute of Standards and Technology's Lisa Carnahan. 



# XML NEWS

When you think of the thousands of legal offices, or any kind of organization with a huge library of files such as WordPerfect or Excel, it's easy to see how appealing i4i's technology could be. DataPipes provide an API that allows them to be connected to any type of application and the documents they produce.

For a collection of word processing documents, file offsets are calculated. For a database file, tuples are used. In a CAD file, vector IDs are used. DataPipes map an arbitrary content model to the data model of the application and manage an ongoing relationship for the user if the file is changed.

We might have included a screen shot with this article, but S4/TEXT is almost invisible to the user. Taking advantage of the COM API in Microsoft Office 97 and 2000, DataPipes will also access Excel and PowerPoint files. i4i has prototyped a DataPipe for Microsoft Access, the goal being one DataPipe for the entire Microsoft Office suite. In fact, a DataPipe can be created for any application with a pub-

lished API. Production DataPipes have been produced for Oracle and Sybase as well as Adobe Acrobat. A CAD prototype using Bentley's MicroStation product has also been developed.

The new version of S4/TEXT uses a MAPI e-mail module that will take everything you need for one document – the Word image, an XML file, a DTD – and distribute it as an e-mail. At the other end, S4/TEXT opens the package for editing, thus permitting collaborative work on an XML project, with XML-enabled change and comment tracking.

"S4/TEXT was the logical solution to solve the USPTO's need to deploy XML to a vast user community," said Michel Vulpe, executive vice president and founder of i4i. "The USPTO is embarking on a system that will be similar to the electronic filing of taxes. About 40% of the world's patents will eventually be processed using this system, which will be a big relief for inventors who don't know if their ideas are protected during the waiting time."

Vulpe began work on his patented technology soon after Congress passed a bill that required the Smithsonian to track every piece in its collection – an inventory of 140 million pieces scattered across 16 museums and galleries, the National Zoo and numerous research facilities around the world. Vulpe was asked to develop methods to allow staff to look at data from different points of view.

In the early '90s he developed the first version of SEMI E36, a standard text exchange for the semiconductor industry. His research showed that 80% of corporate information is held in proprietary formats, and that most documents are re-created about eight times.

This "tyranny of the application" means that documents, CAD files, spreadsheets and databases formatted by different applications can't share data.

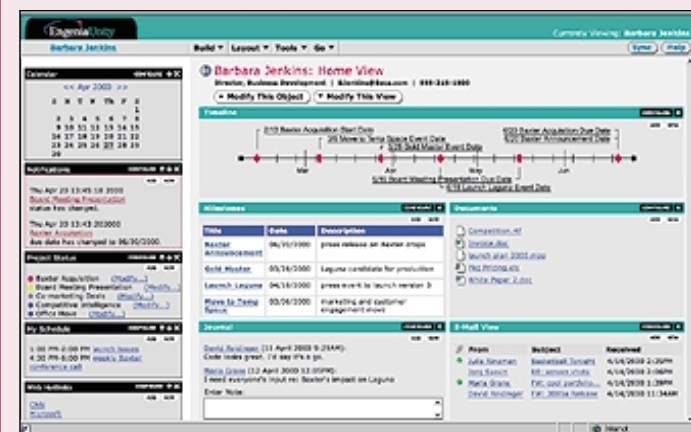
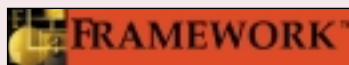
While the benefits of open source software are now well understood, there is a far less known movement called open information (OI). OI sounds a lot like the promise of

## ONE VISION OF UNITY...

...or Those who forget the past are condemned to repeat it. Remember the Xerox Star? Of course you don't. How about Wang OIS? No? Framework ([www.framework.com/](http://www.framework.com/))? The Coordinator? Okay, here's an easy one: Microsoft Office.

Each of these products was based on leading-edge concepts and technologies of the time. And each had the same goal: *office automation*. Now considered a dated term, marketers have replaced it with cool concepts like *knowledge management* and *collaboration*, which is what OA was supposed to accomplish.

No matter. It all comes down to companies selling solutions that claim to help humans cope with other humans. So the interesting part is observing how each generation of OA technology implements the Rodney King Conundrum: "Can't we all just get along?"



Of course, XML means a whole new world of getting along, and Engenia Software, Inc. ([www.engenia.com](http://www.engenia.com)), means to lead the latest charge to greener pastures. Founded by "a highly experienced and recognized team of visionaries from such previous companies as IBM and Lotus," Engenia's first product shows how advanced Lotus Development Corporation's Notes and SmartSuite might be by now if these visionaries hadn't found it necessary to leave.

Engenia Unity integrates with NT 4.0, replacing the file storage and filing paradigm with an XML-based virtual distributed and secure file system that enables information to be shared no matter where it resides. Activities are organized on the Unity desktop by a strategy, an initiative or a project so that every object in the system is unique and in the context of another object or entity. Every object is represented as an XML file that defines the behavior of the object, its relationship to other objects, and other files that may form its user interface or data. Objects in Engenia Unity can inherit behaviors and properties from other objects.



Each user has a selectable, customizable desktop with integrated calendar, e-mail, Web access, news, weather and stock feeds to manage those stock options. For mobile workers synchronization and replication are built into the system, allowing them to work offline as needed.

Following NT's domain model, Unity servers can operate as a peer-to-peer directory service or as a federation of servers sharing directory spaces. Requests from a user's desktop are transparently routed through the federation to the Engenia Unit Service Module (EUSM). If the requested object isn't local, EUSM passes it along to another desktop or server.

So will we all get along? Engenia's Development Kit provides the answer. The IDE enables developers to create and modify objects in something called the WarRoom.

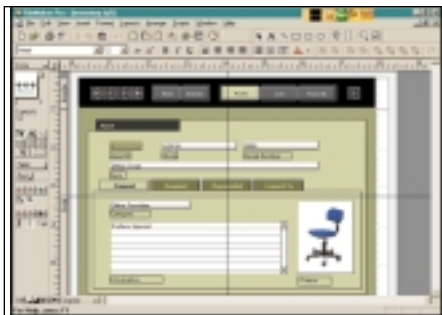
Microsoft's OLE – a document might contain CAD drawing and links to a spreadsheet that are automatically updated as the source changes. But OI implies that the information is open to any application, now and in the future. The genius of i4i's approach is that it allows existing proprietary formats to acquire the properties of open information.

"Open information is absolutely the coming thing," says Vulpe. "Currently, every time we reuse information we create a new document. S4 uses XML/SGML to expose information previously locked in a proprietary format. i4i's philosophy is that a conversion to an OI environment can be done without requiring an enterprise to invest in a whole new system."

Even though XML appears to have a bright future, Vulpe has anticipated the possibility that a more powerful language may emerge in the future. "The main thrust of S4 technology is the architecture," he says. "If there's a new global standard, S4 will adjust." ☉

## FileMaker Aims at Web Developer with XML Support

Boasting new XML support, FileMaker Inc.'s Developer 5 is now available. Aimed at corporate, Web and independent developers, the US\$499 package comes with ODBC and JDBC drivers, giving FileMaker applications potential access to any system supporting XML.



From its origin as a personal database for the Apple Macintosh, FileMaker is aiming to become a significant player in the Web development tool market. With sales of over four million copies and a developer population of 17,000, the subsidiary of Apple Computer, Inc., aims to see FileMaker used in medium-



to high-volume Web sites. To cope with Web traffic, FileMaker Pro 5 Unlimited version (US\$999) achieves scalable load balancing and fault tolerance by using a redundant array of inexpensive computers (RAIC), each running its own copy of the product. ☉

## NEW BOOKS

Neil Bradley's **The XSL Companion; Styling XML Documents** (US\$34.95 ISBN 0-201-67487-4) details the use of Xpath, XSLT and XSL to control and optimize the dynamic formatting of XML documents. Bradley is the author of *The Concise SGML Companion* and *The XML Companion*.

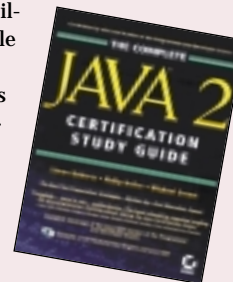
O'Reilly is touting Brett McLaughlin's **Java and XML** (US\$39.95 ISBN 0-596-00016-2) as "arguably the most important new book on Java this year." This is the first book to cover DOM Level 2, SAX 2.0 and JDOM. Chapter 9, "Web-Publishing Frameworks," is online ([www.oreilly.com/catalog/javaxml/chapter/ch09.html](http://www.oreilly.com/catalog/javaxml/chapter/ch09.html)).

CEO Tim O'Reilly took the occasion of his Open Source Convention in Monterey, California, last July to announce **Safari: O'Reilly Books Online** (<http://safari.oreilly.com/>). You pay a



monthly subscription to gain online access to a specific number of titles in the O'Reilly digital library. You can change titles as you wish, and the site provides search, navigation, annotation and bookmarks. Eventually, the entire O'Reilly catalog will be available on Safari.

Sybox has updated its **Complete Java 2 Certification Guide** by Simon Roberts, Philip Heller



and Michael Ernest (US\$49.99 ISBN 0-7821-2825-4). There's new information on the developer exam, hundreds of review and test questions, and coverage of Sun's partner certifications from IBM, Oracle and Netscape.

**The Oracle XML Handbook** by Ben Chang, Mark Scardina, K. Karun, et al., is now available in stores (US\$49.99 ISBN 0-07-212489-X). Written by members of the Oracle XML product development team, the book's CD contains JDeveloper 3.1, Oracle Business Components for Java, Java and Java 2 parsers, and the XSQL Servlet.



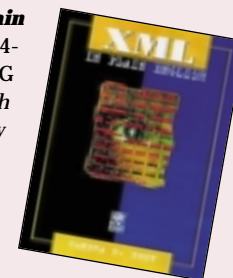
If you're thinking of **Beginning WAP: WML and WMLScript** to develop content for display in mobile devices, then Wrox Press has the book for you (US\$39.99 ISBN 1861004583).

Wouldn't it be great if someone like Sandra Eddy explained **XML in Plain English** (US\$19.99 ISBN 0-7645-4744-5)? This is the second volume of IDG Books' relaunched *In Plain English*

series. The publisher of *XML for Dummies* and *Betty Crocker's cookbooks* also makes amends with Emily Vander Veer's **XML: Your Visual Blueprint for Building Expert Web Pages** (US\$24.99 ISBN 0-7645-3477-7).



Finally, if you hear a distinct metallic thud whenever the info-serf in the adjacent cubicle sits down, MIT Press offers the explanation. **Robo sapiens** is among us (\$29.95 ISBN 0-262133822 <http://mitpress.mit.edu/book-home.tcl?isbn=0262133822>). "This is one of the most mind-stretching – and frightening – books I've ever read," says Sir Arthur C. Clarke. ☉





# XML NEWS

## WWW9 Conference CD-ROM Released

The CD-ROM version of the Conference Proceedings of the Ninth International World Wide Web Conference in Amsterdam, May 15-19, 2000, is now available from Foretec Seminars, Inc. ([www.foretec.com](http://www.foretec.com)), a subsidiary of the Corporation for National Research Initiatives. The proceedings are also available at [www9.org/w9cdrom/index.html](http://www9.org/w9cdrom/index.html).



Cost of the CD-ROM is \$21.21 (includes S&H). Send your check to Foretec Seminars, Inc., 1895 Preston White Dr. #100, Reston, VA 20191. ☎

## Sun Releases Free XSLT Compiler

Sun Microsystems announced the free availability of its new Java 2-based XSLT Compiler at XML DevCon in June ([www.sun.com/xml](http://www.sun.com/xml)). Sun's XSLT compiler creates a Java program that performs only the XSLT transformation instructions associated with a set of XML documents. This "translet" uses only XSLT instructions relevant to these documents, conserving processing resources. Developers can use translets to build XML data conversion into their applications.



Bill Smith, engineering manager of Sun's XML Technology Center, describes the compiler as a "fast, lean Java program that doesn't waste server resources. Since the output of the compiler is so small, developers can now perform transformations on small devices that before now had no ability to transform an XML file." ☎



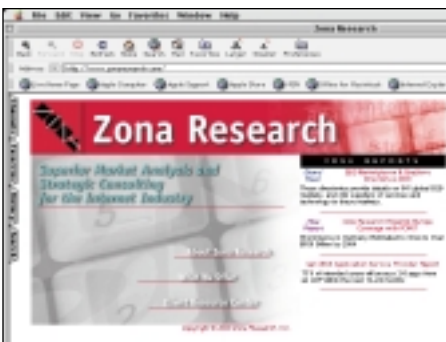
## XHTML, WML Support in CSE HTML Validator

CSE HTML Validator Professional version 4.50 is due for release about the time this issue hits the newsstands ([www.htmlvalidator.com/](http://www.htmlvalidator.com/)). It boasts improved support for XHTML, Wireless Markup Language (WML) and ColdFusion Markup Language (CFML). The award-winning syntax checker switches settings automatically when XHTML documents are detected, and also checks e-mail address syntax and links. A Lite version is available as a free download. ☎



## 2B|^2B In The Zona

Discover a continent, and a cartographer won't be far behind you. Zona Research, Inc. ([www.zonaresearch.com/](http://www.zonaresearch.com/)), offers two unique directories that provide details on 841 global B2B markets, and 242 suppliers of services and technology to these markets. Each entry in the B2B Marketplaces Directory 2000 provides the company name and exchange URL, and a brief description of the function of each exchange. The companion B2B Marketplace Enablers Directories 2000 categorizes 242 B2B providers



of key technologies, infrastructure support and services. Priced at US\$195, the directories are provided as two Excel spreadsheets.

The strange headline above? Oh, that's Extended Backus-Naur Form (EBNF) notation for "to be or not to be." ☎

## XML Developers Collaborate via Oracle

Oracle's new worldwide skills exchange is due for activation September 15, 2000. Using the Oracle Technology Network Exchange (OTN-Xchange <http://otnxchange.oracle.com>), Oracle developers will be able to buy, sell and auction their expertise and manage development projects, many of which will have an XML component.

Modeled after SourceXchange, a collaborative marketplace for financing and managing open source software development, Collab.Net is extending the Oracle Technology Network (<http://technet.oracle.com>) with a section listing jobs, contracts and proposals – not only from OTN but from a number of well-known exchanges, such as FreeAgent.com and techies.com. OTN boasts nearly one million members. Oracle plans to expand the WAP section of the exchange and WAP-enable the site itself.

OTN-Xchange is the third step in a strategy to attract developers. Oracle began with a series of roadshows, then introduced Oracle JDeveloper Release 3.1 and the Oracle XML Developer's Kit (Oracle XDK). The free, downloadable kit includes the Oracle XML Schema Processor. The third step: provide those interested with Oracular opportunities. ☎



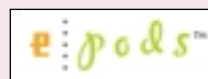
## XML OPENS WINDOWS

BSQUARE Corporation ([www.bsquare.com](http://www.bsquare.com)) has released two XML-based kits that may substantially change the way most people look at Windows CE devices.

CE Transaction Building is an XML-based application development kit for Visual Studio that will allow developers to produce Microsoft BizTalk-enabled applications for Windows CE palmtop devices. BizTalk includes an open design framework for implementing an XML schema and a set of XML tags used in messages sent between applications. With CE Transaction Builder, a developer can take advantage of the BizTalk Framework without necessarily knowing XML. Schemas are downloaded from the BizTalk.org Web site and CE Transaction Builder's Wizards automatically generate components. BSQUARE says its kit enables developers to have a Windows CE application running in a single day.



CE Remote Updater lets Windows CE devices "phone home" for software updates. The kit helps original equipment manufacturers (OEMs) create devices that can update software applications, device data, configuration or even the operating system itself from a remote location. CE Remote Updater uses an XML command file to control the update process and standard HTTP, FTP and CIFS (Common Internet File System) protocols for accessing update files. One of the first products to take advantage of the technology is a portable touchscreen Internet appliance from ePods, Inc. ([www.epods.com/](http://www.epods.com/)). ☎





# JDJ STORE

[www.jdjstore.com](http://www.jdjstore.com)

**JavaCon 2000**  
**p/u from**  
**JDJ5-8**  
**p.104**

# JavaCon 2000

## p/u from

### JDJ5-8

### p.105

**JavaCon 2000**  
**p/u from**  
**JDJ5-8**  
**p.106**

# JavaCon 2000

## p/u from

### JDJ5-8

### p.107



**JavaCon 2000**  
**p/u from**  
**JDJ5-8**  
**p.108**

# JavaCon 2000

## p/u from

### JDJ5-8

### p.109

# JavaCon 2000

## p/u from

### JDJ5-8

### p.110

# JavaCon 2000

## p/u from

### JDJ5-8

### p.111

# JavaCon 2000

## p/u from

### JDJ5-8

### p.112



# JavaCon 2000

## p/u from

### JDJ5-8

### p.113

# JavaCon 2000

## p/u from

### JDJ5-8

### p.114

# JavaCon 2000

## p/u from

### JDJ5-8

### p.115

# JavaCon 2000

## p/u from

### JDJ5-8

### p.116

# JDJ STORE

[www.jdjstore.com](http://www.jdjstore.com)



# Early Innovator... Still Innovating

## Interview... with BRUCE SHARPE



CHIEF TECHNOLOGY OFFICER OF SOFTQUAD SOFTWARE INC.

INTERVIEWED BY RICK ROSS

**XML-J:** *SoftQuad was an early innovator in the world of HTML editing, and has continued – as XML has emerged as a fundamental technology for Web delivery – to innovate new solutions in the XML space. You're coming to it with a perspective – a much longer involvement – than some companies that may only recently have seen the potential value of this kind of tagged extensible technology. Tell us how SoftQuad got into this, where you are today and where you're going.*

**Sharpe:** Our history goes back a long way, right back to the HTML days, and SGML before that. We were among the people who were instrumental in getting those standards in place. And we were the first to bring out a commercial HTML editor. We were cofounders of the W3C, and we've been heavily involved with all that work, including coauthoring the XML spec and being very active in some of the other XML-related working groups. Our director of product technology is chair of the W3C DOM Working Group. We have maintained a lot of presence there.

**XML-J:** *I don't think more than a handful of companies can say that they have been involved with this kind of tagged, structured information delivery for as long as SoftQuad has. Do you have any specific products that are most central to what you are trying to project here at the XML Developer's Conference?*

**Sharpe:** There are three announcements we're making at the show. The first is XMetaL 2.0, the North American launch of the latest version of XMetaL.

**XML-J:** *And what is XMetaL?*

**Sharpe:** XMetaL is our XML content creation solution. It's an easy-to-use, powerful authoring tool for creating XML. The goal of XMetaL is to enable nontechnical people to create valid XML. We want to achieve broad deployment

of XML. We feel that XML content is going to be a key driver in publishing, e-commerce, knowledge management, all kinds of e-business applications, and it's important to enable a lot of people to create XML.

**XML-J:** *So these people will be able to leverage XML as part of whatever it is they're really interested in doing with the technology as a means to an end; not a means to a means.*

**Sharpe:** Yes. We want to put easy-to-use applications in front of the content experts out there who aren't necessarily XML experts. XMetaL is really a platform for creating front ends for XML applications.

**XML-J:** *Would you be creating in this front end? Would a developer create a form-type interface for the manipulation, the content creation? Or is this something that nontechnical end users can do themselves without too much friction?*

**Sharpe:** XMetaL is a very configurable and customizable kind of product. The good news, though, is it doesn't take a high degree of skill to do that kind of customization. We use standard Web technologies like JavaScript, VBScript and CSS for styling, for example. So people who have common Web skills can go a long way in creating user interfaces for specific XML tasks and hide the complexities of XML from their end users.

**XML-J:** *I think people really value that. People have actually begun to make implementation decisions, a little less "putting their toe in the water to see how it feels" and a lot more of "okay, we've decided this will be the architecture we're going to use and we're going to deploy an XML solution." Now they're looking for the vendors, products and services that will come together to deliver enterprise value.*

**Sharpe:** That's exactly right. There are a number of ben-

efits to using XML in all kinds of business applications, and we're definitely seeing momentum there. There's really been a ramping up of the numbers of implementations that have gone beyond just a pilot stage to real deployment. It's happening very rapidly.

**XML-J:** *I think we're seeing another transition, a transition from the Web as the personal publishing platform to the Web as an information-sharing context where XML can be seminal to allowing more peer-to-peer communications, more horizontal communications, than we have seen with just the emergence of HTML alone, which I think was very empowering, but empowering in one-to-many ways. I think XML is creating many-to-many communication opportunities that may impact the use of a tool like your content editor, may create the possibility for dialog to emerge that would not have been reasonable previously. The transaction cost of doing it would have been beyond what a business or marketing person could afford to invest in an aspect of communication. Now, XMetaL is one product. You mentioned there are two other announcements.*

**Sharpe:** We're also announcing our partnership with Documentum and an integration of XMetaL 2.0 with the Documentum 4i eBusiness Edition.

**XML-J:** *Documentum that began at Xerox and is now a much larger thing?*

**Sharpe:** That's right. And they have a very strong offering in the Internet enterprise content management arena. Their XML-oriented products are perfect for Internet-enabled applications and make a tremendous object-based content management system.

**XML-J:** *It sounds like a good partnership.*

**Sharpe:** It's a great partnership. We've been working very closely with them on this, and the result is a very tight integration of XMetaL's content creation abilities with 4i's content management solutions.

**XML-J:** *XMetaL in that context would actually be seamlessly integrated with the overall Documentum service offerings?*

**Sharpe:** Definitely. It's a good example of what I meant when I said XMetaL can provide a front end for XML applications. With the XMetaL integration, an end user gets a convenient view into the Documentum repository, including virtual documents. You can bring XML out of the Documentum repository directly into XMetaL, work with it there and then have it return to the repository.

**XML-J:** *What's the third item?*

**Sharpe:** The third one is another partnership with a company called Extranet. They have great XML training materials and they use XMetaL as a training tool for teaching XML. XMetaL is very friendly for the XML beginner, but also has the depth that lets more experienced people get behind the scenes, underneath it, to the XML. That makes it a perfect tool for teaching people about XML, and Extranet has made it the basis of a number of their offerings.

**XML-J:** So people can relate XML to their own experience, their own business case, their own context, and understand how to use XML through the SoftQuad tool.

**Sharpe:** That's right.

**XML-J:** It sounds like XML can be leveraged in as a component of a solution that somebody else is offering. I mean, you've got an editor that stands alone, but it sounds like with the Documentum and the Extranet cases, XMetaL is functioning as a part of a larger process. Is it something that can be plugged in to, something that our readers may be able to develop for their enterprise customer? Or a solution where XMetaL is actually a modular component of a larger solution?

**Sharpe:** This is actually another of the design goals for XMetaL. We talked about one of them, which is to create very targeted, user-friendly interfaces for end users. Another objective is to make it easy to integrate into any kind of XML application that's out there. It's all COM-based, and there are over 300 COM interfaces. These are all accessible through the scripting I mentioned. XMetaL is a Windows scripting host, which means you can pick the language of your choice.

**XML-J:** How about the Java developer? Does the Java developer access this from any particular approach?

**Sharpe:** There are COM-Java bridges, or in many cases people just create a stand-alone application in Java and interface with XMetaL through the XML itself.

**XML-J:** I think that maybe XMetaL is in a position where it is able to leverage XML's ability to be language agnostic. It doesn't really matter what development language is used for a piece of the business process, if the focus is on the translation of recognized XML standard data at different places in the pipeline. You'll achieve the end goal and the right solution can be applied at the right phase of the pipeline.

**Sharpe:** One of the strengths of XML, of course, is that it is platform neutral. It's a standard we can depend on – it's a great way to transmit information back and forth. XMetaL has all the flexibility to deal with that.

**XML-J:** Is XMetaL available at the SoftQuad site for download?

**Sharpe:** Yes, there is a 30-day trial version of XMetaL available on our site at [www.softquad.com](http://www.softquad.com).

**XML-J:** Are you able and willing to work with smaller developers in a way that they can say, "This will be a cost-effective piece of our solution if we incorporate this large chunk of functionality from a third-party vendor in order to accelerate our time-to-market and be able to deliver a better quality of product at that level than we would have been able to develop ourselves." What's the trade-off in terms of cost? I mean, how does that work out in terms of a cost-benefit evaluation from the other side of the table?

**Sharpe:** There are a number of example solutions and ways for people to get to full solutions from the product, starting with what you get out of the box, which is a pretty rich set of connected applications and customizations, and all the way through to full-service offerings from our partners. We have over 70 partners that can help.

**XML-J:** Is SoftQuad also in a position to provide some integration? And consulting and professional services to help enterprise customers move forward in this way?

**Sharpe:** We have our own professional services group as well as a number of partners.

**XML-J:** Where is SoftQuad located?

**Sharpe:** Our headquarters are in Toronto and our development group is located in Vancouver. We basically relaunched the company about a year and a half ago. We took the company private through a management buyout at that time. We really wanted to focus on the XML opportunities that were there. We went through several rounds of VC financing and then went public earlier this year. We're now a public company traded on the NASDAQ.

**XML-J:** Do you have opportunities for XML-J readers, XML developers? Do you have openings in the technology department? It seems everybody is looking for good developers these days.

**Sharpe:** Yeah, it can be very hard to find good developers. We hang on to the ones we've got for sure. We're growing very rapidly and there are definitely opportunities for anyone interested in something like that in sunny Vancouver.

**XML-J:** Well, there you go, readers. If you're interested in finding a way to do something with somebody who's got staying power in the tagged text management in the structured text field, I think you couldn't find a higher pedigree than SoftQuad's own. It sounds like you guys are on your way to another stage of success. I am excited to hear about it.

**Sharpe:** Well, we're pretty excited too. And I think the show itself that we're at right now is pretty exciting.

**XML-J:** It is exciting, isn't it? It's really great to have someplace where the XML buzz is engaged. You know, everybody is saying it's going to happen. It's obvious to me that it's happening here. Do you feel the same way?

**Sharpe:** Absolutely. People are moving from experimentation to implementation and I think that's great.

**XML-J:** What lies ahead will probably be more exciting and dynamic than what we've seen in the past. I mean, I think the HTML explosion, and the emergence of this first-generation Web in that way, is cool, but to move beyond that into something where it's really, truly dynamically driven, where technologies like yours enable customers to deliver multiparty solutions – multiparty information-distribution Webs – that are able to serve larger and broader audiences....Who are some of your key customers, by the way?

**Sharpe:** We've got quite a number of interesting customers. Amazon.com, for example, uses XMetaL. Other customers include Lucent, British Telecom, IBM, Oracle, Microsoft. USA Today.com is a big customer. Their journalists submit content in XML.

**XML-J:** Instead of phoning this in, they're now actually writing news articles in XML, using XMetaL, and delivering them to the wire line?

**Sharpe:** That's right. So their interviewers can pick it up and can go through the whole process there very efficiently. ☎

Rick Ross is founder of the JavaLobby and president of HeadlineWatch.

[rick@headlinewatch.com](mailto:rick@headlinewatch.com)



“ We want to put easy-to-use applications in front of the content experts out there who aren't necessarily XML experts. XMetaL is really a platform for creating front ends for XML applications ”

# **SYS-CON MEDIA, INC.**

**[www.sys-con.com](http://www.sys-con.com)**



# Don't Miss JDJ's **LINUX** Focus Issue!



[www.JavaDeveloper'sJournal.com](http://www.JavaDeveloper'sJournal.com)

## COMING IN DECEMBER

### Java Developer's Journal Announces Special **LINUX** Focus Issue

Insertion Order Due: **OCTOBER 19, 2000**

Artwork at Our Press: **OCTOBER 26, 2000**

For Advertising Information Contact:

**Carmen Gonzalez**

Vice President, Advertising Sales  
Java Developer's Journal

Call **(201) 802-3021**

or email [carmen@sys-con.com](mailto:carmen@sys-con.com)



**Bluestone Software:**  
**e-Business is OUR Business**

As a leading provider of Web application server technology, enterprise application integration and internet commerce solutions, **Bluestone Software, Inc.'s** cutting-edge technology, multiple career paths and outstanding personal growth have already attracted many of the software industry's best and brightest. Join us as we continue to grow in the 21st century.

**JAVA, XML & E-Business**

- Technical Documentation Specialists
- QA Analysts
- Pre-Sales/System Engineers
- Java Consultants
- Java Product Engineers
- Java Developers/Trainers
- Software Developers
- Web Developers/Designers
- Solutions Architects

All positions require excellent verbal and written communication skills. Development positions require 2+ years of professional experience with JAVA.

Enjoy a casual atmosphere and ongoing training in the latest technology. We offer a competitive salary, comprehensive health benefits, matching 401(k) in addition to a \$1000 bonus through our Cool Cash Referral Program. Send your resume via e-mail to: [jobs@bluestone.com](mailto:jobs@bluestone.com) or mail to: **Bluestone Software, HR Dept., 300 Stevens Dr., Phila., PA 19113-1597** or FAX to (610) 915-5013. EOE.

**Bluestone SOFTWARE**  
Enterprise Interaction Management

Check us out online at [www.bluestone.com](http://www.bluestone.com)

**GLOBAL STRATEGIES**

**LOCAL PARTNERSHIPS**

Experience. Power. Diversity. The strength of a full-service, global IT consulting firm. The convenience of multiple locations to pinpoint both our customers' and consultants' needs. We are Complete Business Solutions, Inc. (CBSI), providing superior business strategies across all industries and throughout the country.

Denver, CO • Los Angeles, CA  
Milpitas, CA • Phoenix, AZ  
Portland, OR • Sacramento, CA  
Seattle, WA

Since 1978, CBSI has focused on supplying state-of-the-art client/server and Internet systems, comprehensive facilities and enterprise network management and structured software engineering. Recognized by Business Week as one of the top 100 growth companies of 1998, our competitive advantage is the result of an incredible team of resources and networked approach.

Over 500 companies nationwide rely on us for expert IT assistance. And it's people like you that will set us apart from the competition. Find out more about CBSI today!

We are currently seeking technical professionals with experience in the following skill sets:

**XML Developers**

In addition to cutting edge opportunities, we offer a full range of excellent benefits, including:

- Premium Paid Health and Dental Plan
- Disability Insurance
- Flexible Spending Account
- Retirement Plan, including a 401(k) Profit Sharing Plan
- Education Assistance and Project-Related Training
- Paid Time Off and Holiday Leave

Empower your career by joining a market leader with amazing growth potential...CBSI. In addition to cutting edge opportunities, we offer a full range of excellent benefits. Please forward your resume to: **CBSI, Attn: Sara Berovic, 1600 NW Compson Drive, Suite 210, Beaverton, OR 97006** or fax to: (503) 748-8901. E-mail: [sberovic@cbisinc.com](mailto:sberovic@cbisinc.com)

**CBSI**  
Complete Business Solutions, Inc.  
[WWW.CBSINC.COM](http://WWW.CBSINC.COM)  
CBSI is proud to be an Equal Opportunity Employer

# Bridging the Management Gap with XML



WRITTEN BY JOHN W. COCULA ]

Managing a company's IT infrastructure has become more complicated in recent years. The tools at the disposal of IT managers haven't quite figured out how to work together to deal with this complexity. Instead, to understand how the infrastructure is actually supporting the business, IT managers struggle to manually assimilate a grab bag of incompatible technologies, often with unsatisfying results.

One possible approach to eliminating the chaos is the common information model (CIM), a standard designed to eliminate the issue of disparate multivendor management platforms. CIM was developed by the Distributed Management Task Force ([www.dmtf.org](http://www.dmtf.org)), a trade group dedicated to the development, adoption and interoperability of management standards and initiatives for desktop, enterprise and Internet environments.

Though infrastructure management vendors such as Tivoli, HP and BMC are migrating their products to be compatible with the CIM standard, it could take a while for this process to play itself out. Until they do, a combination of CIM-based technology and XML offers the best approach to bringing order to enterprise-level IT infrastructure management.

## The Big Picture

The pervasive role of IT infrastructure has become impossible to ignore. While system failures or slowdowns may have barely registered on a company's day-to-day operations in the past, such problems can have much greater consequences today. Internal users may end up without access to critical data, business partners could be shut out of transactions, and Web customers forced to struggle with faltering service may take their business elsewhere.

To avoid such disappointments, business unit managers expect IT to have a complete picture of how a company's IT infrastructure is delivering business services. That leaves IT managers with the unenviable task of relating information from several management domains, such as Internet connectivity, databases and applications; network, system and storage devices; and even non-IT devices like handhelds and satellite communications.

In addition, IT managers need to understand how a resource that a critical business service depends on – for example, a Web server – affects other elements of the corporate information infrastructure. Seemingly discrete, adverse events that impact an individual resource can actually ripple out into much larger corporate problems.

At the moment, however, staying abreast of the various systems feeding into an e-business engine can be a real challenge.

As things stand, individual management systems are primarily event-driven, notifying managers when specific problems occur but offering little in the way of a business context for this information, much less a means to relate information from multiple management tools.

## A Long Wait?

Thankfully, this problem should begin to resolve itself once CIM standards gain wider acceptance.

When CIM-enabled management systems such as HP OpenView, Microsoft's SMS, Novell ManageWise and CA Unicenter are able to share data transparently, IT managers will find it much easier to manage a CIM-only shop as all data will be accessible through a single front end (most likely a Web browser).

CIM has some significant supporters including Microsoft, Cisco and Sun Microsystems. Microsoft, for example, has added CIM support to Windows 2000, and Sun and Cisco have kicked off initiatives rallying developers to the CIM cause.

The reality, however, is that CIM won't solve today's management problems overnight. For one thing, IT managers must cope with a heterogeneous array of management packages – some CIM-enabled, some legacy systems that don't meet CIM standards.

In addition, vendors have some incentive to hold back on the standards adoption process. While open, standards-based software may be the trend on the Internet, the majority of these vendors have made their money selling proprietary systems that sought to displace rather than coexist or integrate with other installed management tools.

It looks like it could be a long wait until the ideal of a smoothly linked, CIM-enabled management environment is realized. To keep the process of enterprise IT management integration moving, industry players need a technology that can bridge the gap between management systems.

## Bridging the Gap

While management vendors work out their implementation issues, we can use XML translation to share information between infrastructure management systems.

According to GartnerGroup, a respected IT research firm, XML-defined data models are the most promising way to integrate corporate applications, including management systems.

XML's content-sensitive nature allows IT managers to pinpoint business problems, rather than simply point out infrastructure events. It's not enough to tell a business unit head that a database fault led to a crippled intranet; it's important for the IT manager to know that sales reps currently can't pull up information on cashmere sweaters.

Our flagship product, Formula, uses XML to correlate IT objects to business objects, making it possible to draw management information from both CIM-enabled and non-CIM-enabled management systems.

While we look forward to the day when infrastructure management systems are CIM-enabled, we believe XML offers an important springboard for companies seeking to integrate today. After all, when it comes to e-business, no one has time to waste. ☎

## AUTHOR BIO

John W. Cocula is the founder and CTO of Managed Objects, a company that makes IT and Internet infrastructure management software.

JOHN@MANAGEDOBJECTS.COM

# **XML GLOBAL TECHNOLOGIES**

**[www.goxml.com](http://www.goxml.com)**



# IBM

[www.ibm.com/developerworks](http://www.ibm.com/developerworks)