

# XML JOURNAL

The World's Leading XML Resource

Volume: 1 Issue: 3

XML-JOURNAL.COM

Announcing... **XML fall DevCon 2000**  
Coming November 12-15, 2000  
**JavaCON 2000** September 24-27, 2000

#### FROM THE EDITOR

**The Show Goes On...**  
by Ajit Sagar pg. 5

#### XML INDUSTRY INSIDER

**Making XML Ready for E-Business**  
by Bob Sutor pg. 22

#### UBIQUITOUS COMPUTING

**XML: A Step Closer to Ubiquitous Computing**  
by John King pg. 28

#### XML & BUSINESS

**Conquering the E-Commerce Landscape with XML**  
by JP Morgenthal pg. 46

#### 2B OR NOT 2B

**Extensibility or Fragmentation?**  
by Coco Jaenicke pg. 56

#### IMIHO

**Staying in Front of Rapid Change**  
by Alan Gold pg. 66

**SYS-CON MEDIA**

Charting recent advances in the XML standards space

## THE EVOLUTION OF XML PROTOCOLS

by Simeon Simeonov  
see page 24

**XML Feature: XML EJBs in Enterprise Application Integration** *E-Ming Tan*  
*XML data is easily available via this EJB component, reducing implementation complexity* 6

**XML and XSL: XSLT Template Modes and Parameters** *Mark Volkmann*  
*Add dynamic attributes to output elements* 10

**XML Feature: XML for C++ Developers** *Ken Blackwell*  
*C++ classes derived from the XML schema definition make your code easier to understand and maintain* 14

**XML Feature: The Evolution of XML Protocols** *Simeon Simeonov*  
*Charting recent advances in the XML standards space* 24

**Java and XML: XML-RPCs and Java** *Israel Hilerio*  
*Remote procedure calls provide easy access to distributed resources* 32

**Information Management: XML-Based Enterprise Information Portals** *Norbert Mikula*  
*Your mainframe on your cell phone* 34

**XML Feature: XML Standards for Customer Information Quality Management** *Ram Kumar*  
*To leverage your strategic information assets you need to handle your metadata correctly* 40

**Objects & XML: UML, MOF and XMI** *Sridhar Iyengar, Ravi Dirckze and Don E. Baisley*  
*A software architecture for the design and implementation of entire distributed object systems* 50

# Soft Quad

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

# OnDisplay

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

# Wrox Press

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

## 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  
 SENIOR EDITOR: JEREMY GEELAN  
 PRODUCTION EDITOR: CHERYL VAN SISE  
 ASSOCIATE EDITOR: NANCY VALENTINE  
 XML INDUSTRY NEWS EDITOR: ALAN WILLIAMSON  
 E-BUSINESS EDITOR: ISRAEL HILERIO  
 JAVA TECHNOLOGY EDITOR: JASON WESTRA

## WRITERS IN THIS ISSUE

DON E. BAISLEY, KEN BLACKWELL, RAVI DIRCKZE, ALAN GOLD,  
 ISRAEL HILERIO, SRIDHAR IVENGAR, COCO JAENICKE, JOHN KING,  
 RAM KUMAR, NORBERT MIKULA, JP MORGENTHAU, AJIT SAGAR,  
 SIMEON SIMEONOV, BOB SUTOR, E-MING TAN

## SUBSCRIPTIONS

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

## SUBSCRIPTION HOTLINE

800 513-7111

COVER PRICE: \$8.99/ISSUE

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

ALL OTHER COUNTRIES \$69.99

(U.S. BANKS OR MONEY ORDERS)

PUBLISHER, PRESIDENT AND CEO: FUAT A. KIRCAALI

VICE PRESIDENT, PRODUCTION: JIM MORGAN

VICE PRESIDENT, MARKETING: CARMEN GONZALEZ

CHIEF FINANCIAL OFFICER: ELI HOROWITZ

ADVERTISING ACCOUNT MANAGERS: MEGAN RING

ROBYN FORMA

JDSTORE.COM: JACLYN REDMOND

AMANDA MOSKOWITZ

ADVERTISING ASSISTANT: CHRISTINE RUSSELL

ADVERTISING INTERN: MATT KREMKAU

GRAPHIC DESIGNERS: ABRAHAM ADDO

JASON KREMKAU

GRAPHIC DESIGN INTERN: AARATHI VENKATARAMAN

WEBMASTER: ROBERT DIAMOND

WEB DESIGNER: STEPHEN KILMURRAY

WEB SERVICES CONSULTANT: BRUNO Y. DECAUDIN

WEB SERVICES INTERN: BRYAN KREMKAU

CUSTOMER SERVICE: ELLEN MOSKOWITZ

## EDITORIAL OFFICES

SYS-CON PUBLICATIONS, INC.

39 E. CENTRAL AVE., PEARL RIVER, NY 10965

TELEPHONE: 914 735-7300 FAX: 914 735-6547

SUBSCRIBE@SYS-CON.COM

XML JOURNAL (ISSN# PENDING)

is published bimonthly (6 times a year) by  
 SYS-CON Publications, Inc., 39 E. Central Ave.,  
 Pearl River, NY 10965-2306

3rd Class Postage rates are paid at  
 Pearl River, NY 10965 and additional mailing offices.

POSTMASTER: Send address changes to:

XML JOURNAL, SYS-CON Publications, Inc.,  
 39 E. Central Ave., Pearl River, NY 10965-2306.

## ©COPYRIGHT

Copyright © 2000 by SYS-CON Publications, Inc. All rights reserved.

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

## WORLD DISTRIBUTION

CURTIS CIRCULATION COMPANY

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

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

SYS-CON MEDIA



WRITTEN BY AJIT SAGAR EDITOR-IN-CHIEF ]

from  
the  
editor  
the  
editor

## The Show Goes On...

Folks, we'll be in the middle of XML DevCon by the time you get this issue. It's strange to describe an event that hasn't happened yet, knowing that your writing will be available as the event is actually taking place. XML DevCon is the largest XML conference ever. The tracks are loaded with content for developers, data architects, marketing managers, business development personnel, evangelists – anyone who's interested in this new and revolutionary technology. I hope that by the time you get this issue some of you are seeing for yourselves the high quality of conferences that SYS-CON plans to provide for the computing community in the years to come.

It's interesting how readily people grasp the concept behind XML. In the past few months I've come across several friends and acquaintances who have asked me about XML-J, what it's about, what XML itself is, and why it's so hot right now. A brief, 10-minute introduction to the concept is all it takes for them to start asking intelligent questions and making suggestions. And these people aren't necessarily in the computer field. In fact, it's easier to explain than HTML, which is limited to browser presentation issues. XML applies to the real world. It's really a language for the business world. It merely finds expression in the world of computers.

Most references to XML that can be found in the computing industry assume Java as the programming language that adds processing capabilities to the data-formatting capabilities offered by XML. However, XML manifests itself in other programming languages too. In fact, this is probably the one standard that both the Microsoft and the Sun camp agree upon, albeit their message for XML's usage is different.

This month we feature an article by Ken Blackwell that talks about how XML translates to C++ objects.

In this issue, as usual, we have a wealth of information on the various facets of XML. E-Ming Tan discusses XML in the world of J2EE's EJB model. Mark Volkmann brings you an interesting article about XSL template modes. Norbert Mikula talks about XML and Enterprise Information Portals, a popular area in which XML is being applied in B2B applications. You might see Mr. Mikula at XML DevCon 2000.

XML allows us to specify data formats. Object languages such as Java allow us to add behavior to the data represented by XML to allow it to be processed. If applications ran on a single machine, this would be sufficient to write business applications. However, in a distributed world data needs to be transported. Data expressed in XML can be transported over a variety of object communication protocols. Simeon Simeonov's article on XML protocols describes the different modes of transport available to data in XML format.

An obvious area in which XML finds expression is the world of customer relationship management. After all, XML is a great way to express relationships, given its hierarchical nature. Ram Kumar brings you an interesting article on how CRM applications can leverage XML.

Of course, in addition to all these insightful articles, we offer up-to-date information and news from our expert group of regular columnists. Like the previous two issues, this is an information-packed issue that will inform you about the latest developments in the XML community.

The expansion of XML applications into different spheres of business has one unfortunate side effect – chaos. Data definitions expressed in XML are emerging in all facets of the industry. Soon people will be asking, "Whose XML is it anyway?" Does your "payment" mean the same as mine? Did your "airline" DTD include a choice for "air-conditioning control"? Hopefully, over the next few years, emerging standards facilitated by industry consortiums will help solve this problem. ☺



AJIT @SYS-CON.COM

## AUTHOR BIO

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

# **XML**

## **ENTERPRISE JAVABEANS IN ENTERPRISE APPLICATION INTEGRATION**

XML data is  
easily available  
via this EJB  
component,  
reducing  
implementation  
complexity



A few years back, when I was at Singapore Telecom, I remember being asked why I was so excited about Java. My answer was that it's cross-platform. In return I got a suspicious look, which was fair: Java was new then.

Now XML is the hottest thing on the planet and the situation is a bit different. With Microsoft (as well as those anti-Microsoft folks) backing up this technology, I suspect nobody will ever give me that kind of look again.

Microsoft argues that Sun's cross-platform Java strategy is flawed because of the enormous amount of retraining needed. Microsoft sees XML as the ultimate solution not only in EAI, but even inter-enterprise, with B2B. Simple Object Access Protocol (SOAP) submitted to IETF is one such effort. However, until this initiative receives worldwide acceptance and becomes fully implemented, EAI and e-commerce will still have sunny days with technologies like Java and CORBA.

Although there are serious issues involved in EAI that would probably take a book to cover, this article is for you if, during the course of your EAI strategic planning, you come across the following:

- Common, open-format document storage
- An easy, standard-data retrieval method from the stored document, with no SQL involved
- A fully documented validating facility against any type of predefined document structure
- A configurable software component that provides transparent persistency for the document with no SQL codes to write
- A cross-platform solution

The solution discussed below combines the flexibility of XML with the cross-platform, component-based qualities of Enterprise JavaBeans.

## Objective

The initial design goal is to provide a helper class to isolate the *n*-tier application from the complexities of maintaining an XML DOM object, yet still be able to tap into the power of XML. Container-managed entity beans, with few convenient remote methods, have been developed to store an XML document and to retrieve its elements and attributes as text.

## Overview

The solution presented here addresses EAI using one of many popular ways – XML with EJB. I chose XML because it's an open-standard format, and Java for the usual popular reason. The solution consists of a sample Java client and an XML-powered EJB component (hereafter called the XML bean). With regard to functionality, the XML bean exposed remote interface methods for the client (be it a stand-alone application, cgi/servlet or even another EJB) to parse an XML document, store it as persistent data and retrieve the XML data easily. The best thing about this solution is that it's simple and straightforward to configure and use. Hopefully this will encourage you to keep reading and finally deploy an XML EJB for your EAI!

## Implementation

I deployed a container-managed JDBC EJB running on the popular server BEA WebLogic 4.5. Apache Software Foundation's XML parser Xerces 1.0.3 and Xalan 0.20.0, formerly contributed by IBM, were used as the XML processing engine. The leading RDBMS, Oracle 8 database, was used to provide XML data persistency. The sample codes included later in this article assume you're running Windows 9X/NT to connect to the Oracle 8 database. But the script files included for compiling the XML bean and its client can be easily modified for different platform and installation directories.

For simplicity's sake, a container-managed JDBC entity bean was developed as the XML bean. It exposes three business methods. One is basically an application-specific EJB create() method, and the other two are retrieval functions. The retrieval methods allow you to extract the entire XML document, individual elements or attributes from the XML bean.

## Database Configuration

Because the XML bean uses a variable-length character string to store the XML document, different data schema may be needed if the back end is not

Oracle. Please consult your DBA regarding this matter. In the Oracle database environment, create a table called ejbXMLBeans with two columns called "id" of type varchar2 and "xmldata" of type long. For example:

```
create table ejbXMLBeans (id varchar2(5) constraint PK_SML_ID primary key, xmldata long)
```

Map the XML bean's attributes, xmldataId and xmldata, to the table created by including these lines in the deployment descriptor:

```
(j dbc
  tableName          ejbXMLBeans
  dbIsShared         false
  poolName           xmlEJBPool
  (attributeMap
    ; EJBean attribute      Database column name
    ; -----
    xmldataId              id
    xmldata                 xmldata
  ); end attributeMap
); end jdbc
```

Don't forget to set up a connection pool called XMLJBP. Please refer to BEA WebLogic documentation for details. Finally, deploy the XML bean and you're all set!

## Remote Interface Methods Review

Let's review the business methods of the XML bean – there are four of them – before we launch the sample client application.

### 1. public XMLImpl findByPrimaryKey(XMLImpl PK primaryKey)

This is the only finder method of the XML bean. It's declared in the home interface as XMLImplHome. The finder method locates the XML bean based on a unique five-character ID. If a bean is found, it returns the remote interface of the bean, XMLImpl, to an EJB client. Otherwise it's returned as invalid.

### 2. public XMLImpl create(String xmldataId, String newXmldata)

The XML bean has a single application-specific create method, declared in the home interface XMLImplHome. This method allows the XML bean client to store the XML document – XMLData – with a five-character primary key, xmldataId. It returns a remote interface, XMLImpl, of the XML bean to an EJB client. If the XML document isn't well formed, or it's not a valid document against a declared DTD, the XML bean will throw a ProcessingErrorException exception to the EJB client.

### 3. String getXMLData(String xpath)

The first business method of the XML bean, declared in the remote interface XMLImpl, is to retrieve the text of an element/attribute based on its XPath (which we'll cover shortly). If the xpath specified is invalid or there's no match, it simply returns "".

### 4. o String getXMLDoc()

The last business method of the XML bean is to return the entire XML document. It's declared in the remote interface XMLImpl. This is simply a convenient method that's provided in case there's a need to retrieve the whole content of the XML document. It's not that useful, though, since the clients must be able to parse the XML document tree by itself, which defeats the purpose of using the XML bean in the first place. All four methods are defined in XMLBean, the XML bean class. A more elegant design should have made the getXMLDoc() and getXMLData(..) methods inside a session bean, away from the XML entity bean. This will associate the XML entity bean solely with the database-related operation, while the session bean deals with application-specific requests.

## A Little Background on XPath

After seeing so many X's roaming around in EAI and e-commerce, you might be wondering what on earth XPath is. XPath is a standard defined by W3C to retrieve XML element/attribute data. You may have a better idea of implementing a way to retrieve data from an XML document handled by the XML bean, but the goal here isn't to reinvent the wheel if there's already such a thing as XPath in the XML world.

To make a long XPath specification short so you won't fall asleep, XPath models an XML document as a tree of nodes, easily referred to and retrieved using an expression similar to a UNIX path convention. There are different types of nodes, including element nodes, attribute nodes and text nodes. XPath defines a way to compute a string value for each type of node. In this article XML bean wholeheartedly borrows this expression syntax. Not only that, it even borrows the XPathAPI class supplied by Apache Software Foundation for this purpose. I understand that XPathAPI will be included as part of Xerces' parser core class in the future, but until then this class has to be included separately as XML bean's package.

## Sample XML Bean Application

I still believe that an example is worth a thousand times more than a discussion, so I've included a sample e-commerce order constructed as an XML document, together with its DTD, for this purpose. The XML document is as follows:

```
<?xml version="1.0"?>
<!DOCTYPE page SYSTEM "e:\personal\ejb\dworl d\xml\ejb\order.dtd">
<page type="ViewOrderStatusData">
  <orderid>888</orderid>
  <customer type="1">
    <first-name>E-Ming</first-name>
    <last-name>Tan</last-name>
    <email>futurewave@iname.com</email>
  </customer>
  <storeid>s0003</storeid>
  <sessionid>1689</sessionid>
  <notes>
    <note>This is an order data.</note>
  </notes>
</page>
```

Note that the sample XML document includes a DTD in its "<!DOCTYPE" tag. This will force the XML bean to automatically validate the document structure against the DTD. If there isn't a DTD specified in an XML document that's passed to the XML bean, no document validation will be carried out by the bean but it is well formed.

The next thing to show you with this XML document is how to invoke the sample client program that's included in the XML bean to process it. Type the following at the command prompt (assuming that you've unzipped the EJB files into directory e:\personal\ejb and you're running the WebLogic server application locally):

```
java -classpath %CLASSPATH%;. dworl d.xml.ej b.Client t3://l ocal -
host: 7001 e:\personal\ejb\dworl d\xml\ejb\order.xml 10022
```

The client will pass the order.xml file to the WebLogic server, and the EJB container will create an XML entity bean with a primary key "10022" to process the XML document. The XML bean will check the validity and form of the document. If the check passes, the XML bean accepts the document and saves it into the datastore. Otherwise it throws a processing exception to the client and refuses to accept the document.

Running the client again with an extra command-line argument (XPath expression, "/page/orderid/text()"), as shown below, will show you that the client retrieves the order ID from the XML document.

```
java -classpath %CLASSPATH%;. dworl d.xml.ej b.Client t3://l ocal -
host: 7001 e:\personal\ejb\dworl d\xml\ejb\order.xml 10022
"/page/orderid/text()"
```

You'll notice that the XML bean returns "888" to the client application. Another example is to run the client as follows:

```
java -classpath %CLASSPATH%;. dworl d.xml.ej b.Client t3://l ocal -
host: 7001 e:\personal\ejb\dworl d\xml\ejb\order.xml 10023
```

```
java -classpath %CLASSPATH%;. dworl d.xml.ej b.Client t3://l ocal -
host: 7001 e:\personal\ejb\dworl d\xml\ejb\order.xml 10023
"/page/customer[@type='1']/first-name/text()"
```

This will return "E-Ming" : -)

Simple, right? If the answer is yes, the XML bean has achieved its design goal. To learn more about XPath, check out the references at the end of this article.

## Assumption

This article assumes that a data consumer application is aware of the availability of the XML data. It's up to you whether to use technologies like JMS or other MOM solutions to make notification of the new XML data. You can also create a server program, or modify the service broker program (see my article, "The Java-CORBA Way to Large-Scale Software Design," *JDI*, Vol. 4, issue 8) for that purpose. That's beyond the scope of this article, however.

The XML bean facilitates convenient enterprise data-sharing across multiple applications, with common data converted into XML. The XML data is easily available via this EJB component, reducing implementation complexity. Even though the XML bean avoids client-side XML processing, and until somebody has a better idea, the client still needs to know how to construct a well-formed and valid XML document before calling the EJB. In addition, the consumer of the XML document has to know the structure of data in order to specify its XPath, yet there's no way of modifying the XML document once it's been accepted by the XML bean.

A last warning: XML and XPath are still evolving, so some of the API used by the XML bean may be subject to changes in the future. You should visit <http://xml.apache.org> often, in case there are API updates of Xerces and Xalan.

## Conclusion

XML has unlimited applications. It's stunning to realize how it has evolved into a metalanguage that no longer serves as an entity that separates data from its presentation layer or as some sort of common format in EAI. It's become more than that. Some people will argue that XML is slow, since everything is plain text. But Java was "slow" too - especially during its early years!

## Acknowledgments

Special thanks to Alex Lee Chiew Kwooi and Yong Nyit Wah for their help in preparing this article. ☺

## References

1. **Extensible Markup Language (XML):** [www.w3.org/XML/](http://www.w3.org/XML/)
2. **XML Path Language (XPath) Version 1.0 W3C** - Proposed Recommendation 8 October 1999: [www.w3.org/TR/1999/PR-xpath-19991008](http://www.w3.org/TR/1999/PR-xpath-19991008)
3. **Apache XML Project:** <http://xml.apache.org>
4. **Using XML With WebLogic Server:** [www.weblogic.com/docs50/class-docs/xml.html](http://www.weblogic.com/docs50/class-docs/xml.html)
5. **1999 XML News:** <http://metalab.unc.edu/xml/news1999.html>

## AUTHOR BIO

E-Ming Tan is a senior e-business consultant based in MacLean, Virginia. He was the creator of a new paging application and an architect of the Web-enabling implementation of the application in Java during the Web's early days.





# Info Shark

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



*Add dynamic attributes to output elements  
to create HTML links and anchors*

# XSLT Template Modes and Parameters

The eXtensible Stylesheet Language (XSL) supports the transformation and formatting of XML data for presentations. When used for transformations, XML input data is transformed into HTML, different XML and many other kinds of text output. This article focuses on specific capabilities of the transformation portion of XSL, namely, template modes and parameters.

XSL stylesheets are composed of templates (although the latest recommendation also supports stylesheets without explicit templates). These templates are selected using pattern matching. An XSL processor begins by trying to

find a template that matches “/”, the document root. Directives in that template (<xsl:apply-templates>) can cause instantiation of other templates for specific kinds of nodes in the input XML document. When a template is instantiated, its content is output. Often this content is a combination of HTML tags and XSL elements that extract data from the input XML document.

There are two basic approaches to performing transformations: data driven and template driven. In the template-driven approach there are fewer templates and they tend to resemble a typical “computer program” with conditional (<xsl:if> and <xsl:choose>) and looping (<xsl:for-each>) control structures. This style of writing stylesheets may be intuitive for software developers but not for typical Web-content developers.

For those who'd rather not create stylesheets that resemble computer programs, the data-driven approach is more attractive. There are more templates in this style, perhaps one for each type of node in the input XML document. This makes it easy to focus attention on how nodes of a particular type should be output within each template. It also results in templates that are easier to understand.

This brings us to the reason for “modes.” Modes allow for the creation of multiple templates that output the same node type many different ways. Each template can have an associated mode name. Using <xsl:apply-templates> causes templates to be evaluated, and it can specify that only templates with a given mode name should be considered.

For example, when transforming an XML document that describes CDs in a music collection, sometimes the output should merely be its title. At other times the output should be the title, year and a table describing all the tracks on the CD (see Figure 1).

Another way to vary the output for a given node type is to pass parameters to a template. The template can use <xsl:if> and <xsl:choose> to test the values of the parameters in order to decide what to output. Parameter values can also be output. Another option is to use parameter values as the values of HTML “style” attributes to specify formatting.

In Listings 1–3 an XML document that describes a music collection is transformed using an XSL stylesheet. The resulting HTML has two sections. The first is a table of contents that lists alphabetically the artists and CDs in the collection. Each item in the table of contents serves as a link to more detailed information about the CDs in the second section.

The template that matches “/” creates the main structure of the HTML output. Since title text is used in two places – for the title tag in the head section and a header tag (h1) in the body section – it's convenient to place that text in a variable and get its value in the appropriate places. Inside, the body tag <xsl:apply-templates> is used to instantiate the template for the root element, “music-collection.”

The template that matches “music-collection” creates the table of contents. For the artist and CD lists, <xsl:apply-templates> is used to instantiate templates that output the appropriate information. At the end of the music-collection template <xsl:apply-templates> is used again to output detailed information about the CDs of each artist. Modes are used to distinguish between the tem-



FIGURE 1 XSL output display

plates used for the table of contents and those used for the more detailed information that follows it. Templates with a mode of summary are used only for the table of contents. Templates without a mode are used for the detailed output.

When entries in the table of contents are clicked, the display scrolls to the corresponding information. This is achieved by creating HTML anchor tags. Anchors with a "name" attribute mark spots within the output HTML document. Anchors with an "href" attribute create links to those marked spots. The names of the links are created dynamically based on the names of artists and the titles of CDs. To create tags that require dynamic attributes, <xsl:attribute> can be used. This allows the name and value of each dynamic attribute to be specified. Another method, the one used here, is to use attribute value templates (AVTs). These are strings that contain {Xpath expressions}. The anchor

name for artist anchors is simply the artist's name. The anchor name for CD anchors is a concatenation of the artist's name and the CD title. This is necessary for cases in which multiple artists have a CD with the same title.

In the artist template with no mode a variable created with the name "even" indicates whether the current artist is an even-numbered artist when they're sorted alphabetically. This variable is passed to the CD template with no mode; there it's used to determine the background color of the tables that describe tracks on the CD: orange for even-numbered artists, yellow for odd-numbered. The CD template parameter "even" has a default value of false(), which is used if that template is instantiated without passing the parameter. As you might guess, false() is a function that returns the boolean value for false.

In the CD template without a mode a variable named "has-times" is created

that indicates whether any of the tracks of the CD have a time specified. This variable is used within the CD template to determine whether the track table should contain a third column for times. It's also passed to the track template where it's used to determine whether times should be output. The track template parameter "has-times" has a default value of false() that's used if that template is instantiated without passing the parameter.

Listings 1-3 demonstrate several important XSLT concepts, including use of template modes, template parameters, sorting, variables, conditional tests and outputting HTML tables. They also demonstrate how to add dynamic attributes to output elements – in this case to create HTML links and anchors.

## AUTHOR BIO

Mark Volkmann holds an MS in computer science and has been working in object-oriented software engineering since 1993. He provides consulting and training in Java and XML for Object Computing, Inc., in St. Louis, Missouri.

### LISTING 1 MUSIC-COLLECTION.DTD

```
<!ELEMENT music-collection (owner, artist*)>
<!ELEMENT artist (name, cd*)>
<!ELEMENT cd (title, track*)>
<!ELEMENT track (name, time?)>

<!ELEMENT name (#PCDATA)>
<!ELEMENT owner (#PCDATA)>
<!ELEMENT time (#PCDATA)>
<!ELEMENT title (#PCDATA)>

<ATTLIST cd year CDATA #IMPLIED>
```

### LISTING 2 MUSIC-COLLECTION.XML

```
<?xml version="1.0"?>
<!DOCTYPE music-collection SYSTEM
  "music-collection.dtd">
<?xml-stylesheet
  type="text/xsl" href="music-collection.xsl"?>

<!-- Only three tracks per CD are provided
to limit the amount of XSL output. -->
<music-collection>
  <owner>Mark Volkmann</owner>

  <artist>
    <name>Throwing Muses</name>

    <cd category="alternative" year="1986">
      <title>Throwing Muses</title>
      <track><name>Call Me</name></track>
      <track><name>Green</name></track>
      <track><name>Hate My Way</name></track>
    </cd>

    <cd category="alternative" year="1988">
      <title>House Tornado</title>
      <track><name>Col der</name></track>
      <track><name>Mexican Women</name></track>
      <track><name>The River</name></track>
    </cd>

    <cd category="alternative" year="1989">
      <title>Hunkpapa</title>
      <track><name>Bea</name></track>
      <track><name>Dizzy</name></track>
      <track><name>No Parachutes</name></track>
    </cd>

    <cd category="alternative" year="1991">
      <title>The Real Ramona</title>
```

```
<track>
  <name>Counting Backwards</name>
</track>
<track><name>Hi m Danci ng</name></track>
<track><name>Red Shoes</name></track>
</cd>
```

```
<cd category="alternative" year="1992">
  <title>Red Heaven</title>
  <track>
    <name>Furi ous</name>
    <time>3: 52</time>
  </track>
  <track>
    <name>Fi repi le</name>
    <time>3: 11</time>
  </track>
  <track>
    <name>Di o</name>
    <time>2: 51</time>
  </track>
</cd>
```

```
<cd category="alternative" year="1995">
  <title>Uni versi ty</title>
  <track>
    <name>Bri ght Yellow Gun</name>
    <time>3: 43</time>
  </track>
  <track>
    <name>Start</name>
    <time>2: 48</time>
  </track>
  <track>
    <name>Hazi ng</name>
    <time>3: 15</time>
  </track>
</cd>
```

```
<cd category="alternative" year="1996">
  <title>Li mbo</title>
  <track><name>Buzz</name></track>
  <track>
    <name>Ruthi e's Knocki ng</name>
  </track>
  <track><name>Frel oader</name></track>
</cd>
</artist>
```

```
<artist>
  <name>Breeders</name>
```

```

<cd category="alternative" year="1990">
  <title>Pod</title>
  <track><name>Glorious</name></track>
  <track><name>Doe</name></track>
  <track><name>Oh!</name></track>
</cd>

<cd category="alternative" year="1993">
  <title>Last Splash</title>
  <track><name>New Year</name></track>
  <track><name>Cannonball</name></track>
  <track><name>Invisible Man</name></track>
</cd>
</artist>

<artist>
  <name>Belly</name>

  <!--cd category="alternative" year="1993">
    <title>Star</title>
    <track><name>Angel</name></track>
    <track><name>Dusted</name></track>
    <track><name>Every Word</name></track>
  </cd-->

  <cd category="alternative" year="1995">
    <title>King</title>
    <track>
      <name>Seal My Fate</name>
      <time>3: 35</time>
    </track>
    <track>
      <name>Red</name>
      <time>4: 00</time>
    </track>
    <track>
      <name>Silverfish</name>
      <time>4: 25</time>
    </track>
  </cd>
</artist>

<artist>
  <name>Donelly, Tanya</name>

  <cd category="alternative" year="1997">
    <title>Lovesongs For Underdogs</title>
    <track><name>Pretty Deep</name></track>
    <track>
      <name>The Bright Light</name>
    </track>
    <track><name>Landspeed Song</name></track>
  </cd>
</artist>

<artist>
  <name>Hersh, Kristin</name>

  <cd category="alternative" year="1994">
    <title>Hips and Makers</title>
    <track><name>Your Ghost</name></track>
    <track><name>Beestung</name></track>
    <track><name>Teeth</name></track>
  </cd>
</artist>
</music-collection>
<?xml version="1.0"?>

```

### LISTING 3 MUSIC-COLLECTION.XSL

```

<!-- This XSL stylesheet demonstrates usage of
template modes, template parameters,
sorting, conditional tests, variables,
and adding dynamic attributes to
output elements. -->
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:output method="html"/>

  <xsl:template match="/">
    <xsl:variable name="title">
      <xsl:value-of
        select="/music-collection/owner"/>'s
        CD Collection
      </xsl:variable>

    <html>

```

```

    <head>
      <title>
        <xsl:value-of select="$title"/>
      </title>
    </head>
    <body>
      <h1><xsl:value-of select="$title"/></h1>
      <xsl:apply-templates
        select="music-collection"/>
    </body>
  </html>
</xsl:template>

<xsl:template match="music-collection">
  <!-- Output a table of contents for
  artists and CDs. -->
  <table border="2" cellpadding="5px">
    <!-- Output the table heading row. -->
    <tr>
      <th>Artists</th>
      <th>CDs</th>
    </tr>

    <!-- Output a single table data row. -->
    <tr>
      <!-- Output a table of contents for
      artists sorted by name. -->
      <td valign="top">
        <!-- There are two templates that
        output artists. They are
        distinguished by their "mode". -->
        <xsl:apply-templates
          select="artist" mode="summary">
          <xsl:sort select="name"/>
        </xsl:apply-templates>
      </td>

      <!-- Output a table of contents for
      CDs sorted by title. -->
      <td valign="top">
        <!-- There are two templates that
        output cds. They are
        distinguished by their "mode". -->
        <xsl:apply-templates
          select="artist/cd" mode="summary">
          <xsl:sort select="title"/>
        </xsl:apply-templates>
      </td>
    </tr>
  </table>

  <!-- Output detailed information about
  each CD sorted by name. -->
  <xsl:apply-templates select="artist">
    <xsl:sort select="name"/>
  </xsl:apply-templates>
</xsl:template>

<!-- Summarized information about artists -->
<xsl:template match="artist" mode="summary">
  <div>
    <!-- Link to detailed artist information
    in the output document. -->
    <a href="#{name}">
      <xsl:attribute name="href">#<xsl:value-of
        select="name"/></xsl:attribute>
      <xsl:value-of select="name"/>
    </a>
  </div>
</xsl:template>

<!-- Detailed information about artists
(no mode) -->
<xsl:template match="artist">
  <!-- Determine whether this is an
  even-numbered artist. -->
  <xsl:variable name="even"
    select="(position() mod 2) = 0"/>

  <!-- Create an anchor for the
  detailed artist information. -->
  <a name="{name}">
    <xsl:attribute name="name">
      <xsl:value-of select="name"/>
    </xsl:attribute>
    <h2 style="color: red">
      <xsl:value-of select="name"/>

```

```

</h2>
</a>

<!-- Output detailed information about
each of the CDs by this artist
sorted by descending year. -->
<xsl:apply-templates select="cd">
  <xsl:sort select="@year"
order="descending"/>
  <xsl:with-param name="even"
select="$even"/>
</xsl:apply-templates>
</xsl:template>

<!-- Summarized information about CDs
-->
<xsl:template match="cd" mode="summary">
  <div>
    <!-- Link to detailed CD information
in the output document. -->
    <a href="#{../name}-{title}">
      <xsl:attribute
name="href">#{xsl:value-of
select=".. /name"/}<xsl:value-of
select="title"/></xsl:attribute>
      <xsl:value-of select="title"/>
    </a>

    <!-- Follow each CD title with
the name of the artist. -->
    - <xsl:value-of select=".. /name"/>
  </div>
</xsl:template>

<!-- Detailed information about CDs
(no mode) -->
<xsl:template match="cd">
  <!-- This parameter indicates
whether the CD is from an even-numbered
artist -->
  <xsl:param name="even"
select="false()"/>

  <!-- The table background color is
based on whether the CD is
from an even or odd-numbered
artist. -->
  <xsl:variable name="bgcol or">
    <xsl:choose>
      <xsl:when
test="$even">orange</xsl:when>
      <xsl:otherwise>yellow</xsl:otherwise>
    </xsl:choose>
  </xsl:variable>

  <!-- Determine whether the CD has
times for any of its tracks.
-->
  <xsl:variable name="has-times"
select="count(track/time) > 0"/>

  <div style="margin-left: 2em">
    <!-- Create an anchor for the
detailed CD information. -->
    <a name="{../name}-{title}">
      <xsl:attribute name="name">
        <xsl:value-of
select=".. /name"/><xsl:value-of
select="title"/>
      </xsl:attribute>
    <div
style="margin-top: 2ex; font-
size: medium">
      <xsl:value-of
select="title"/> -
      <xsl:value-of
select="@year"/>
    </div>
  </a>

  <!-- Only output a table containing
track information if this CD
has data for at least one
track. -->
  <xsl:if test="count(track) > 0">
    <table border="2"

```

```

    <table border="2"
      <xsl:attribute name="bgcol or"="{ $bgcol or}">
        <!-- Output the table head-
ing row. -->
        <tr>
          <th>Track #</th>
          <th>Name</th>
          <!-- Only output a column
for times if the
current CD has at
least one track wi th
a time. -->
          <xsl:if test="$has-times">
            <th>Time</th>
          </xsl:if>
        </tr>

        <!-- Output the table data rows. -->
        <xsl:apply-templates
select="track">
          <xsl:with-param name="has-
times"
select="$has-times"/>
        </xsl:apply-templates>
      </table>
    </xsl:if>
  </div>
</xsl:template>

<!-- Detailed information about
tracks -->
<xsl:template match="track">
  <!-- This parameter indicates
whether any track of the CD
has a time. -->
  <xsl:param name="has-times"
select="false()"/>

  <!-- Output a table row for this
track. -->
  <tr>
    <!-- Output the track number. -->
    <td align="right">
      <xsl:value-of select="posi-
tion()"/>
    </td>

    <!-- Output the track name. -->
    <td><xsl:value-of
select="name"/></td>

    <!-- Only output this column if
the current CD has at least
one track with a time. -->
    <xsl:if test="$has-times">
      <td>
        <!-- At least one track of
the current CD has a
time but this track may
not. -->
        <xsl:choose>
          <xsl:when test="time">
            <xsl:value-of
select="time"/>
          </xsl:when>
          <xsl:otherwise>
            <!-- A &nbsp; forces
table borders
to be drawn. -->
            &#160;
          </xsl:otherwise>
        </xsl:choose>
      </td>
    </xsl:if>
  </tr>
</xsl:template>
</xsl:stylesheet>

```



# GCA 1/3

www.gca.com



The background of the cover is a vibrant blue with various technical illustrations. There are several thick, curved lines in silver and red that sweep across the frame. In the lower-left, there are interlocking blue gears. A network diagram with blue squares and circles connected by lines is visible in the bottom-left corner. A large, 3D blue network connector with multiple pins is positioned in the center-right. The overall aesthetic is high-tech and digital.

XML FEATURE

# XML FOR C++ DEVELOPERS

C++ classes  
derived from the  
XML schema  
definition make  
your code easier  
to understand  
and maintain

To newcomers to the XML world, it might seem as if XML and Java are somehow connected at the hip. There are certainly synergies between the two technologies, largely because they've come of age at the same time. Consequently, many of the new developments in XML are first implemented in Java, and we're now seeing new Java developments leveraging the standardization of XML. In the real world, however, most new code is still written in C++ and often involves interaction with existing applications.

## XML-Enabling Existing Applications

Many organizations now face the task of XML-enabling existing applications as quickly as possible. Some of these projects are trying to achieve better application integration, while others are just trying to achieve buzzword compliance. Whatever the motivation, you'll find that most of the information you read in journals and online sources tends to be directed toward enabling existing applications for XML with minimal changes to existing source code.

It's important to keep in mind that when I use terms like *old* or *legacy*, I don't necessarily mean those half-million lines of 20-year-old COBOL in the accounting system back-end. For many developers the "old" code that now requires an XML interface is more likely the ERP system that finally got rolled out last summer. Odds are, the application and associated integration components are written in an object-oriented language like C++ or Java.

A variety of techniques are being used to XML-enable existing code, but the common thread among all of them is the translation of XML messages into some other format that's already understood by the application. For instance, to XML-enable your ERP system you might employ an adapter that translates XML documents to SAP's iDoc format (see Figure 1). Another approach might be to translate XML messages into EDI transactions using a middleware broker (see Figure 2). The fundamental goal of these approaches is to meld XML capabilities onto existing applications without changing the applications themselves. If packaged products aren't available to perform the translation, you can always develop your own conversion logic using open standards like DOM, SAX, XSL, XT and others.

In this first wave of the XML-enabling of existing applications, Java has carved out a significant market. You'll find that most of the XML brokering products are implemented in Java. At their cores are data format filters that convert XML to and from other data formats, including other dialects of XML.

In the short term, data format translations offer a quick solution to providing XML support in existing applications. In the long term, however, they have the potential to significantly add to the complexity of the overall system. That translates into increased cost of ownership and decreased performance.

## Integration vs Translation

If the first generation of XML proliferation is to XML-enable existing applications, then the second must be to add XML support to applications that produce and consume XML data, not translate and forward it. This is a logical next step since adding integral XML support to these applications makes many of the data translation issues disappear. Figure 3 depicts a fully XML-enabled enterprise. In this environment XML is not only the data format used for communication with external entities via the Internet, it's also the native format for enterprise application interoperability.

What does it mean to add integral XML support? In layman's terms it means anticipating and designing for XML rather than adding XML as an extension of the application after the fact. It means propagating schema data types through to the application rather than doing data type conversions during an XML import/export process. And it means mirroring

the XML document structure into the class structure of the application in a friendly and object-oriented way.

It's in meeting these types of requirements that standards like DOM and SAX start to fall short. These are great tools if you're doing XML data translation because they provide a generic interface to any XML document. They're independent of the problem domain of whatever application will ultimately process the data encapsulated in the XML document. But if you're adding integral XML support to an application, the last thing you want for your application data is a generic API. It's much more productive to have an object-oriented class structure that directly represents the data and methods defining your problem domain. These classes can then have methods for generating or parsing compliant XML documents. With this type of solution you can spend time solving your business problems instead of dealing with the intricacies of detailed XML programming.

## Shortcomings of DOM

Okay, so I've made some accusations against DOM (and SAX). Now I'll try to back up my assault on this standard. Before I do, let me say for the record that the only reason I'm not also describing the advantages of DOM is because I consider them to be well documented and readily available in publications like this one and through online resources. There's plenty of information floating around that will help you understand where DOM is appropriate, but not much that will help you figure out where it isn't.

By far the most significant shortcoming of DOM, one that makes life hard for C++ programmers, is that it fails to leverage the underlying principle of XML: "meaning, not markup." XML tags not only delineate data elements, they also provide an organization for the data by logically encapsulating related and derivative data items. XML schema designers invest a lot of time and brainpower in their designs, and mapping the XML document structure to C++ OOP concepts will reuse this intellectual capital.

DOM, on the other hand, provides a generic API that has no meaning in relation to any specific document structure. Its API is designed with XML document-processing tasks in mind, such as validation or translation. Also, DOM doesn't allow you to access XML data elements using the data type information available in XML schema standards. All XML data is presented to the calling program as string data, which makes DOM more susceptible to cast errors, constraint errors and simple typos when comparing string literals.

The schema for a simplistic purchase order in Listing 1 provides an example of these limitations. Listing 2 gives the code necessary to populate a DOM structure for generating valid XML for the purchase order schema in Listing 1.

Notice first that most of the code in Listing 2 deals with XML-isms, such as creating and manipulating nodes, setting attributes and managing child lists. These interfaces aren't relevant to the problem domain (i.e., issuing a purchase order) and greatly increase the complexity of the code.

Since the DOM interfaces aren't related to the XML document structure, you'll notice the use of string literals throughout Listing 2. For example, the following code creates the Seller element:

```
DOM_Node Seller = PODOC.createElement("Seller");
```

The element tag is set with the string literal "Seller" and is therefore susceptible to typos in the code. Compiling the code from Listing 2 provides virtually no error detection in the context of the purchase order DTD that defines the intended XML output.

Finally, the DOM parser used in Listing 2 provides its own classes for Unicode support, namely DOMString. While a pretty powerful class, it doesn't integrate well with string classes from other frameworks such as std::string from STL or CString from MFC.

## C++ Classes for XML

Implementing a DTD or schema with C++ classes eliminates most of the problems associated with DOM usability. C++ classes can map directly to the XML document structure, thereby providing a context relevant interface to the XML data. By deriving the C++ class hierarchy from the XML

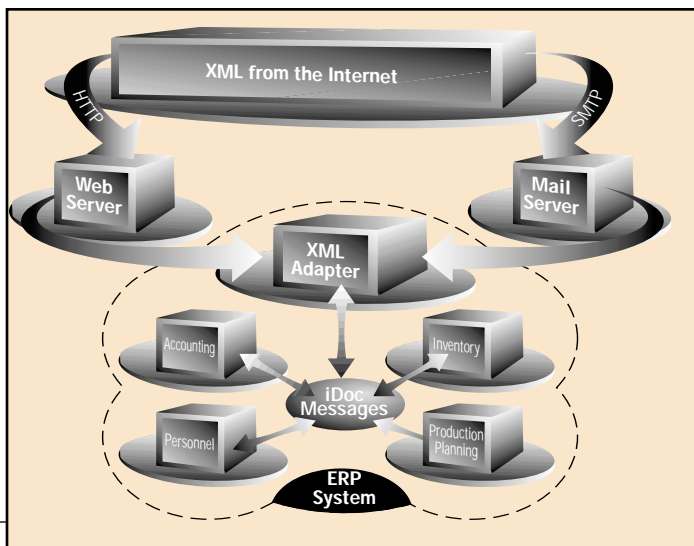


FIGURE 1 XML adapter approach

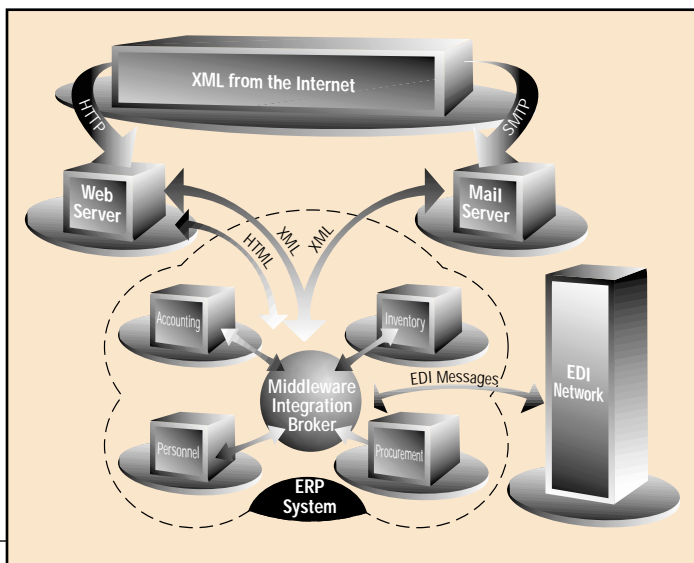


FIGURE 2 XML-enabled middleware approach

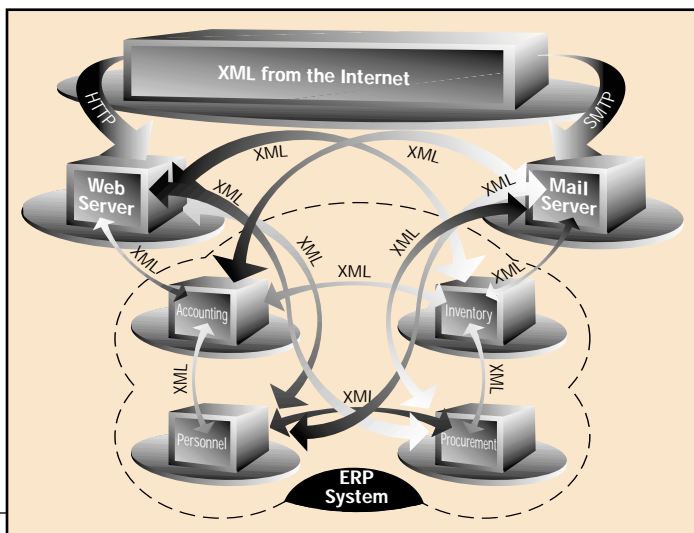


FIGURE 3 XML-enabled enterprise

structure, you reuse the work of the schema designers and provide a common language between the schema designers and the developers. Of course, for this to work well, the schema designers need to consult with the programmers to make sure their schemas lend themselves to proper OOP concepts such as inheritance, encapsulation, strict data typing and more. My experience is that a good schema design makes for a good class hierarchy and vice versa, so having these people collaborate is a win for everyone.

Another advantage of the C++ classes approach is easier integration with the rest of the application. For example, most C++ DOM parsers include their own libraries for handling code page translation for internationalization. While these libraries may perform well, most likely they're not the same libraries you may be using in the rest of your application. If you're coding for Windows NT, for instance, you're likely using the native NLS APIs provided by Windows NT. Trying to use the native APIs in your main application, while simultaneously using a third-party library in the DOM parser, can cause integration problems if the same set of code pages isn't supported by each package. Integration is also easier with C++ classes that are written using the same application framework (such as MFC or STL libraries) as the rest of your application.

Since DOM is defined as an interface, not an implementation, the ability to do derivations of DOM classes varies greatly by implementation. In general, derivation of DOM classes is difficult because the DOM API was designed with parsing XML documents in mind, resulting in many read-only interfaces. If you have application-specific data associated with XML elements, such as a database key, DOM doesn't make it easy to associate this data with DOM nodes. With schema-derived C++ classes, however, you can quickly derive classes from the schema-derived base classes and encapsulate your application data.

Compare Listing 3, which generates XML output via schema-derived C++ classes, to Listing 2, which uses DOM. You'll see that the code in Listing 3 is more readable and better encapsulated, and has better data type checking (both compile and runtime) than the DOM code in Listing 2. Most important, interfaces are specific to the problem domain of the application rather than XML-ese.

As I mentioned above, the quality of C++ classes derived from an XML schema is directly proportional to the quality of the schema itself. You can do things in a schema that will wreak havoc on the classes. The good news is that these pitfalls are also bad schema design in themselves, irrespective of the C++ integration.

It's important that you make use of the new draft XML schema standard as soon as it's practical to do so. C++ classes derived from DTD files lack data typing and other advanced features available in XML schema. If you're working in a Windows-only environment, then the schema standard supported by the MSXML parser from Microsoft, commonly known as XML-Data Reduced (XDR), is a viable alternative until the new XML schema standard matures. As the XML schema standard evolves with the introduction of custom data types, data constraints and inheritance, the mapping of schemas to C++ classes will be more complete.

## Summary

In this article we've discussed the benefits of creating custom C++ classes for XML generation as opposed to generic interfaces such as DOM or SAX. C++ classes derived from the XML schema definition provide a more object-oriented approach and will make your code easier to understand and maintain. Your code won't be littered with XML logic that's tangential to the business problems you're attempting to solve.

This focus of this article has been on generating XML data from C++ classes, not parsing XML input data. My next article will describe a unique methodology for parsing XML data in C++ classes, which will provide all of the object-oriented benefits described here and increased performance compared to traditional XML parsers. ☘

## AUTHOR BIO

Ken Blackwell is the chief technical officer of Bristol Technology, Inc., where he oversees product architecture and research in XML, middleware and transaction analysis technologies.

KENB@BRISTOL.COM

# Sequoia

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

# LISTING 1

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

<schema name="posample.xsd"
  xmlns="http://www.w3.org/1999/05/06-xml schema-
    1/structures.xsd">
  <elementType name="PurchaseOrder" model="open">
    <sequence>
      <elementTypeRef name="Buyer"/>
      <elementTypeRef name="Seller"/>
      <elementTypeRef name="Items"
        minOccurs="1"
        maxOccurs="*" />
    </sequence>
    <attrDecl name="PONumber" required="true">
      <datatypeRef name="number"/>
    </attrDecl>
    <attrDecl name="Date" required="true">
      <datatypeRef name="date"/>
    </attrDecl>
  </elementType>

  <attrDecl name="PONumber" required="true">
    <datatypeRef name="number"/>
  </attrDecl>

  <attrDecl name="Date" required="true">
    <datatypeRef name="date"/>
  </attrDecl>

  <elementType name="CompanyName" model="open">
    <datatypeRef name="string"/>
  </elementType>

  <elementType name="BuyingAgent" model="open">
    <datatypeRef name="string"/>
  </elementType>

  <elementType name="Address" model="open">
    <datatypeRef name="string"/>
  </elementType>

  <elementType name="City" model="open">
    <datatypeRef name="string"/>
  </elementType>

  <elementType name="State" model="open">
    <datatypeRef name="string"/>
  </elementType>

  <elementType name="Zip" model="open">
    <datatypeRef name="string"/>
  </elementType>

  <elementType name="Buyer" model="open">
    <sequence>
      <elementTypeRef name="BuyingAgent"/>
      <elementTypeRef name="CompanyName"/>
      <elementTypeRef name="Address"/>
      <elementTypeRef name="City"/>
      <elementTypeRef name="State"/>
      <elementTypeRef name="Zip"/>
    </sequence>
  </elementType>

  <elementType name="Seller" model="open">
    <sequence>
      <elementTypeRef name="CompanyName"/>
      <elementTypeRef name="Address"/>
      <elementTypeRef name="City"/>
      <elementTypeRef name="State"/>
      <elementTypeRef name="Zip"/>
    </sequence>
  </elementType>

  <elementType name="Items" model="open">
    <datatypeRef name="string"/>
    <attrDecl name="PartNumber" required="true">
      <datatypeRef name="string"/>
    </attrDecl>
```

```
<attrDecl name="Quantity" required="true">
  <datatypeRef name="integer"/>
</attrDecl>
<attrDecl name="Price" required="true">
  <datatypeRef name="real"/>
</attrDecl>
</elementType>

<attrDecl name="PartNumber" required="true">
  <datatypeRef name="string"/>
</attrDecl>

<attrDecl name="Quantity" required="true">
  <datatypeRef name="integer"/>
</attrDecl>

<attrDecl name="Price" required="true">
  <datatypeRef name="real"/>
</attrDecl>

</schema>
```

# LISTING 2

```
// -----
// Includes
// -----

#include <util/PlatformUtils.hpp>
#include <parsers/DOMParser.hpp>
#include <dom/DOM_Node.hpp>
#include <dom/DOM_NamedNodeMap.hpp>
#include "DOMTreeErrorReporter.hpp"
#include <string.h>
#include <stdlib.h>

// -----
//
// main
//
// -----

struct {
  char *PartNumber;
  char *Quantity;
  char *Price; // Yuck, DOM can't handle numbers!
} items[] = {
  "123456-7890", "4", "10.20",
  "234567-8901", "5", "23.21",
  "098765-4321", "2", "98.42",
  NULL, NULL, NULL
};

int main(int argc, char* argv[])
{
  // Initialize the XML4C2 system
  try
  {
    XMLPlatformUtils::Initialize();
  }

  catch(const XMLException& toCatch)
  {
    cerr << "Error during Xerces-c Initialization.\n"
      << "Exception message:"
      << DOMString(toCatch.getMessage()) << endl;
    return 1;
  }

  // create a DOM Document node
  DOM_Document PODoc = DOM_Document::createDocument();
  DOM_Text tab = PODoc.createTextNode(DOMString("\t"));

  // create the root node for a PO
  DOM_Node PO = PODoc.createElement("PurchaseOrder");
  PODoc.appendChild(PO);
```



```

// set the PONumber
DOM_Attr PONumber = PODoc.createAttribute("PONumber");
PONumber.setValue(DOMString("12345")); // strings only, yuck!
DOM_NamedNodeMap attrs = PO.getAttributes();
attrs.setNamedItem(PONumber);

// set the PODate
DOM_Attr PODate = PODoc.createAttribute("Date");
PODate.setValue(DOMString("March 30, 2000")); // strings
only, yuck!
attrs.setNamedItem(PODate);

//
// create the Buyer Info
//
DOM_Node Buyer = PODoc.createElement("Buyer");
PO.appendChild(Buyer);

// set the Buyer subelements
// first, the agent's name
DOM_Node BuyingAgent =
PODoc.createElement(DOMString("BuyingAgent"));
DOM_Text BuyingAgentName = PODoc.createTextNode(DOM-
String("J. Q. Buyer"));
Buyer.appendChild(BuyingAgent);
BuyingAgent.appendChild(BuyingAgentName);

// next, the buyer's company
DOM_Node BuyerCompany =
PODoc.createElement(DOMString("CompanyName"));
DOM_Text BuyerCompanyName = PODoc.createTextNode(DOM-
String("Acme Buying, Inc.));
BuyerCompany.appendChild(BuyerCompanyName);
Buyer.appendChild(BuyerCompany);

// next, the buyer's address
DOM_Node BuyerAddress =
PODoc.createElement(DOMString("Address"));
DOM_Text BuyerAddressName = PODoc.createTextNode(DOM-

```

```

String("123 21st Street"));
BuyerAddress.appendChild(BuyerAddressName);
Buyer.appendChild(BuyerAddress);

```

```

// next, the buyer's city
DOM_Node BuyerCity =
PODoc.createElement(DOMString("City"));
DOM_Text BuyerCityName = PODoc.createTextNode(DOM-
String("New York"));
BuyerCity.appendChild(BuyerCityName);
Buyer.appendChild(BuyerCity);

```

```

// next, the buyer's state
DOM_Node BuyerState =
PODoc.createElement(DOMString("State"));
DOM_Text BuyerStateName = PODoc.createTextNode(DOM-
String("NY"));
BuyerState.appendChild(BuyerStateName);
Buyer.appendChild(BuyerState);

```

```

// finally, the buyer's zip
DOM_Node BuyerZip =
PODoc.createElement(DOMString("Zip"));
DOM_Text BuyerZipName = PODoc.createTextNode(DOM-
String("01234"));
BuyerZip.appendChild(BuyerZipName);
Buyer.appendChild(BuyerZip);

```

```

//
// create the Seller Info
//
DOM_Node Seller = PODoc.createElement("Seller");
PO.appendChild(Seller);

```

```

// set the Seller subelements
// first, the Seller's company
DOM_Node SellerCompany =
PODoc.createElement(DOMString("CompanyName"));
DOM_Text SellerCompanyName = PODoc.createTextNode(DOM-

```

# Ixiasoft

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

# Go Online and Subscribe Today!

[www.SYS-CON.com](http://www.SYS-CON.com)



or

Call 1-800-513-7111

Subscribe to the Finest Technical  
Journals in the Industry!

# GET YOUR OWN!

**SYS-CON**  
MEDIA

```
String("Zeta Selling, Inc.");
SellerCompany.appendChild(SellerCompanyName);
Seller.appendChild(SellerCompany);

// next, the seller's address
DOM_Node SellerAddress =
PODoc.createElement(DOMString("Address"));
DOM_Text SellerAddressName = PODoc.createTextNode(DOM-
String("123 12th Street"));
SellerAddress.appendChild(SellerAddressName);
Seller.appendChild(SellerAddress);
```

```
// next, the seller's city
DOM_Node SellerCity =
PODoc.createElement(DOMString("City"));
DOM_Text SellerCityName = PODoc.createTextNode(DOM-
String("Los Angeles"));
SellerCity.appendChild(SellerCityName);
Seller.appendChild(SellerCity);
```

```
// next, the seller's state
DOM_Node SellerState =
PODoc.createElement(DOMString("State"));
DOM_Text SellerStateName = PODoc.createTextNode(DOM-
String("CA"));
SellerState.appendChild(SellerStateName);
Seller.appendChild(SellerState);
```

```
// finally, the seller's zip
DOM_Node SellerZip =
PODoc.createElement(DOMString("Zip"));
DOM_Text SellerZipName = PODoc.createTextNode(DOM-
String("43210"));
SellerZip.appendChild(SellerZipName);
Seller.appendChild(SellerZip);
```

```
//
// Create an Item element for each item on the PO
//
```

```
for (int i = 0; items[i].PartNumber; i++) {
    // create an item node
    DOM_Node Item = PODoc.createElement("Item");
    PO.appendChild(Item);
```

```
    DOM_NamedNodeMap attrs = Item.getAttributes();
```

```
    // set the PartNumber
    DOM_Attr PartNumber = PODoc.createAttribute("PartNum-
ber");
    PartNumber.setValue(DOMString(items[i].PartNumber));
    attrs.setNamedItem(PartNumber);
```

```
    // set the Quantity
    DOM_Attr Quantity = PODoc.createAttribute("Quantity");
    Quantity.setValue(DOMString(items[i].Quantity));
    attrs.setNamedItem(Quantity);
```

```
    // set the Price
    DOM_Attr Price = PODoc.createAttribute("Price");
    Price.setValue(DOMString(items[i].Price));
    attrs.setNamedItem(Price);
}
```

```
// write out the XML
cout << PODoc << endl;
```

```
//
// The DOM document and its contents are reference
// counted, and need no explicit deletion.
//
return
```

## LISTING 3

```
void CMainFrame::OnExportXML()
{
    // create the PO object and set attributes
    posample::PurchaseOrder PO;
    PO.Set_Date("March 28, 2000");
    PO.Set_PONumber(12345);
```

```

// set Buyer info
posample::Buyer Buyer = PO.Get_Buyer();
Buyer.Set_BuyingAgent("J. Q. Buyer");
Buyer.Set_CompanyName("Buyer, Inc.");
Buyer.Set_Address("123 21st Street");
Buyer.Set_Ci ty("New York");
Buyer.Set_State("NY");
Buyer.Set_Zi p("12345");
PO.Set_Buyer(Buyer);

// set Seller info
posample::Seller Seller = PO.Get_Seller();
Seller.Set_CompanyName("Seller, Inc.");
Seller.Set_Address("123 21st Street");
Seller.Set_Ci ty("New York");
Seller.Set_State("NY");
Seller.Set_Zi p("12345");
PO.Set_Seller(Seller);

// XList is a derivative of the MFC CList template
eXactML::XList<posample::Item *> & itemLi st =
PO.Get_i temLi st();

for (int i = 0; items[i].PartNumber; i++) {
    posample::Item *pltem = new posample::Item;
    pltem->Set_PartNumber(items[i].PartNumber);
    pltem->Set_Quanti ty(items[i].Quanti ty);
    pltem->Set_Pri ce(items[i].Pri ce);

    itemLi st.AddTail(pltem);
}

// verify that we have valid XML
try {
    PO.IsValid();
}
catch (eXactML::XException e) {
    CString msg;
    msg.Format("Exception: (%s) while validating

```

```

imported XML", e.GetMsg().c_str());
AfxMessageBox("XML import validation failed!\n" +
msg, MB_OK|MB_ICONSTOP);
return;
}

// export the data to XML format.
try {
    PO.Emi tXMLToFi le("posample.xml");
}
catch (CFileException e) {
    TCHAR szCause[255];
    e.GetErrorMessage(szCause, 255);
    AfxMessageBox("CFileException: " +
    CString(szCause));
}
catch (XException e) {
    CString msg;
    msg.Format("Exception: (%s) while generating
XML", e.GetMsg().c_str());
    AfxMessageBox("Export to " +
    CString("posample.xml") + " failed!" + "\n" + msg);
    return;
}
catch (CException e) {
    TCHAR szCause[255];
    e.GetErrorMessage(szCause, 255);
    AfxMessageBox("CFileException: " +
    CString(szCause));
}
catch (...) {
    AfxMessageBox("Unrecognized exception while gener-
ating XML.");
    return;
}
}

```



# Computer Work

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



*Why XML Schema will make e-business applications more robust and easier to maintain*

# Making XML Ready for E-Business

This month I'm going to discuss a new technology being standardized within the World Wide Web Consortium (W3C) that will have a major impact on the way XML will be used in the next few years.

As many of you know, Document Type Definitions (DTDs) were part of the XML 1.0 specification and provided a way to define and constrain the structure of a document. By *document* here I'm not restricting us to books or articles. Rather I mean the full class of data that XML might represent – including application-to-application messages, program data and anything else that needs the structure that XML provides. DTDs can provide only some of the structure needed for e-business applications and that's why the forthcoming W3C XML Schema definition is so important.

The XML DTD specification is essentially a subset of the SGML DTD specification originally standardized in 1974 under the leadership of Charles Goldfarb. SGML in turn grew out of the GML work started by Goldfarb and others at IBM in the 1960s. In some sense, then, XML is a member of at least the third generation of markup languages – which is why, even though it became a W3C Recommendation only in early 1998, we can state that it's a mature technology.

## Shortcomings of DTDs

All this maturity aside, though, much (but not all) of the work involving SGML was for publishing applications. While XML DTDs allow you to define elements and attributes and specify how elements should be nested, you can't say anything about the form of the element content and very little about attribute content. If you're concerned only with publishing, this is adequate, but it's too simple if you're using XML as the syntax for portable application data.

Consider the following XML snippet:

```
<stockTransaction type="buy">  
  <symbol>IBM</symbol>  
  <quantity>1000</quantity>  
  <limitPrice  
    currency="US">120.00</limitPrice>  
</stockTransaction>
```

If we want to create a visual rendition of this we can generate HTML and produce a table as shown in Table 1.

While it's important for this information to be correct, we're not checking the format of the values in any way, just displaying them. A DTD would allow us to specify that the only allowable values for the attribute *type* are "buy" and "sell," but it can't express that the stock symbol must be from a given collection, that the quantity must be a positive integer or that the limit price must be a decimal within a certain range of values. It's up to the application program that's processing the data to ensure that the values are acceptable. If the application is going to compute with these values or do database queries based on them, they need to be in the correct form. It's up to every application program that processes the data to ensure its validity before using it. Thus every application programmer must write or borrow routines that check that the data is in the correct format.

TRANSACTION TYPE	SYMBOL	QUANTITY	LIMIT PRICE
buy	IBM	1000	US \$120.00

TABLE 1 Table using HTML

## AUTHOR BIO

Bob Sutor is IBM's program director for XML technology and the chief strategy officer of OASIS, the Organization for the Advancement of Structured Information Standards ([www.oasis-open.org](http://www.oasis-open.org)).



## Moving Beyond DTDs

The above examples are fairly straightforward. But what if we're expecting a product number in the form of "3 capital letters, followed by two digits, followed by a hyphen, followed by 4 digits" – for example, *FGY78-5427*? This is the basic problem with DTDs: they force applications to implement data validity checks for a wide range of potential formats, an error-prone process and one that must be repeated for every programming language that might process the data. The XML parser checks that the document is well formed and perhaps does a structural validity check, but the application program does the rest.

It makes more sense to move the format-checking routine into the parser so that applications can concentrate more on doing whatever they're supposed to do with the data. XML parsers are generic tools that work on XML data from any source. If there is common XML processing performed on data before an application does its special work, it's reasonable to consider moving that common function into the parser.

Of course, we could end up with some pretty big parsers if we don't build them in a modular way that allows us to choose the functionality we want. That's why, for example, IBM's XML4J parser (now the primary code-base for the Apache Xerces XML parser) offers a validating configuration along with a smaller nonvalidating version. Nevertheless, moving this functionality out of applications and into the parsers is a smart thing to do to get more reliable code.

Let's look at how we might constrain the quantity element in the transaction above. If we simply want to insist that it be a positive integer, we can express it this way in the XML Schema definition:

```
<element name="quantity" type="positiveInteger"/>
```

If we want to limit it to a maximum of 10,000 shares it takes a bit more work, but we can do it like this:

```
<element name="quantity">
  <simpl eType base="positiveInteger">
    <maxExclusive value="10001">
  </simpl eType>
</element>
```

## Using the XML Schema

While you could write these by hand, XML tools that support schema creation will have user interfaces that simplify the way you define the format for the XML data. The draft XML Schema specification has a whole document devoted to datatypes. David Fallside's Primer (*XML Schema Part 0: Primer*, by David C. Fallside of IBM), also part of the draft specification, provides an excellent introduction.

By the way, you probably noticed that XML Schema uses XML syntax. Schemas can thus be manipulated by standard XML tools such as editors and XSLT processors.

I've made the point that XML application programming becomes easier when datatype checking occurs in the parser. However, a subtler point is that the schema actually documents the data formats. There's no standard way of doing it in a DTD, though you can play tricks with attributes and comments.

The schema allows us to separate the processing logic between the parser and the application. If we later decide that we really want to restrict the quantity of stocks to be bought or sold to be less than 5,000, we can simply update the schema and all programs that use the new schema will inherit the change. Obviously this is easier than requiring changes to all the programs themselves.

The types used in XML Schema go beyond the simple integers and decimals in the foregoing example. The `stockTransaction` is an instantiation of a complex type. XML Schema provides sophisticated methods for reusing schemas and the simple and complex types within them. For example, we could create a new type by extending `stockTransaction` to include additional information such as the stock owner name, brokerage, brokerage ID number, date of transaction, settlement date and so on. (This kind of derivation should be familiar to C++ and Java programmers.) If the base `stockTransaction` gets changed, the new complex types created from it will automatically inherit the changes.

You can also restrict types so that only a subset of the possible values can be used in the XML data. We could, for example, create a special kind of stock transaction where the quantity must be between 100 and 1,000 and the limit price must be greater than \$100.

These features in XML Schema have given XML enough power to represent real business data for transactions, application and session data, and database support. The work done by the W3C XML Schema working group has added an important component for making XML ready for e-business. XML is now a first class language for representing portable data that is independent of the programming language, application and operating system used to create it.

## Conclusion

Later this year you should start looking for tools that support XML Schema but you should plan your migration strategy from DTDs now. For the basic parser technology I recommend you learn about and track the work being done on Xerces within the Apache organization at [xml.apache.org](http://xml.apache.org). You can get the latest XML Schema specifications from the W3C at [www.w3.org/tr](http://www.w3.org/tr). ☛



SUTOR@US.IBM.COM

# GCA 1/3

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





XML FEATURE

# THE EVOLUTION OF XML PROTOCOLS

Charting  
recent advances  
in the XML  
standards space



*ML protocols can be broadly classified into two generations.*

*First-generation protocols are based purely on XML 1.0. Second-generation protocols take advantage of two revolutionary XML standards – XML Namespaces and XML Schema. This article analyzes the reasons why we need to make a shift to second-generation protocols, and looks at industry activity in this area.*

## First-Generation XML Protocols

Generally speaking, protocols specify in detail how certain business/application/network services are accessed through a set of requests and how responses/replies are received. XML protocols' requests and responses are encoded in XML.

There were many interesting first-generation protocol efforts. They informed the XML community of important protocol requirements and particular approaches to satisfy these requirements. Unfortunately, few of the first-generation XML protocols achieved multivendor support and broad adoption. I mentioned one of them, Web Distributed Data Exchange (WDDX), in my article on B2B computing in the last issue (*XML-J*, Vol. 1, issue 2). Another one worth mentioning is XML-RPC.

XML-RPC is a remote procedure call protocol. RPCs identify the target procedure to be called, pass some parameters to it and receive a response. XML-RPC uses HTTP as the underlying transport protocol, but all call and response data is in XML. Listing 1 shows an XML-RPC call example.

First-generation XML protocols share many common characteristics. Generally they use a fixed set of "envelope" elements for specifying requests and responses. XML-RPC uses `methodCall`, `methodName` and `methodResponse`. They also use a fixed set of elements to identify data types. In Listing 1 these are `i4` and `string`.

First-generation protocols also share common problems.

### EXTENSIBILITY THROUGH NAMESPACES

First-generation protocols weren't very extensible. The protocol architects had to reach agreement before any changes were implemented and the protocol version had to be revved up to let tools distinguish new protocol versions from old ones and handle the XML appropriately. For example, when XML-RPC and WDDX added support for binary data, both protocols had to update their specifications, and the protocol implementations on all different languages and platforms supporting the protocols had to be updated.

Needless to say, the overhead of constantly revising specifications and deploying updated tools for handling the latest versions of the protocols imposes limits on the speed and scope of adoption of first-generation protocols. Don't get me wrong: protocols like XML-RPC and WDDX do very useful things and are great at what they do. But it would be practically impossible to use them as the base for the One Generic XML Protocol to End All Protocols. The main issue is that the facilities of XML 1.0 on their own offer little help in creating extensible XML protocols. XML DTDs aren't designed with this in mind.

The extensibility problem boils down to the need to decentralize the evolution of XML protocol specifications. For example, TrustMe.com wants to add security support to an existing protocol. The company will define some XML format (schema) for representing security information that it will need to mix in with the existing protocol schema. How can it do so without any ambiguity?

In early 1998 the XML community, having realized the general importance of this problem, started work on Namespaces in XML. The original W3C Note stated: "We envision applications of XML in which a document instance may contain markup defined in multiple schemas. These

schemas may have been authored independently. One motivation for this is that writing good schemas is hard, so it is beneficial to reuse parts from existing, well-designed schemas....These considerations require that document constructs should have universal names whose scope extends beyond their containing document." The effort became a W3C Recommendation in early 1999.

The namespace approach is simple and elegant. While a full discussion is beyond the scope of this article, a single example will give you a sense of the power of namespaces. There are two parts to using namespaces: you first identify them via URIs, then you identify the use of the namespace via a name prefix (see Listing 2). That's all it takes to integrate the work of TrustMe.com.

Because of their ability to promote the decentralized evolution and reuse of XML specifications and applications, namespaces can be considered the most fundamental advance in the XML standards space since XML 1.0.

### DATA TYPES AND VALIDATION THROUGH SCHEMAS

Without namespaces, most first-generation XML protocols stuck to a single DTD to describe the representation of serialized data in XML. Most implemented support for simple data types (strings, numbers, booleans and date-time values) as well as structured types (arrays, associative arrays, a k a structures and tabular data). In general, first-generation XML protocols used just a few XML elements. This made building tools supporting these protocols relatively easy. Everything seemed fine until people realized they'd lost the ability to declaratively specify what the protocols were working with.

Consider a simple example about people's names and ages. We can represent this information as:

```
<person>
  <name>Peter Smit</name>
  <age>42</age>
</person>
```

Say we want to represent the data in WDDX so that it can be freely exchanged between applications written in different programming languages and running on different platforms. We can do this as:

```
<struct>
  <var name='name' >
    <string>Peter Smit</string>
  </var>
  <var name='age' >
    <number>42</number>
  </var>
</struct>
```

So far, so good. We can process this chunk of WDDX to our heart's content.

But wait a minute. How do we know that the WDDX data we are processing has to do with people's names and ages? For example, what if we get:

```
<struct>
  <var name='name' >
    <string>Fast Server</string>
  </var>
  <var name='price' >
    <number>1500</number>
  </var>
</struct>
```

This chunk of XML is also valid WDDX according to the WDDX DTD. However, it's most definitely not about people. It most likely has to do with hardware products. Another way to represent it using generic XML could have been:

```
<product>
  <name>Fast Server</name>
  <price>1500</price>
</product>
```

By now you must see what the problem is. It has to do with the fact that, by choosing to express your information using the single schema of an XML protocol, you've lost the ability to automatically perform validation on the data you receive. A validating XML processor will clearly be able to distinguish between `<product>` and `<person>`. This isn't the case once the information is represented in a common data format such as WDDX.

Is this a big problem? It depends on your viewpoint. I think it makes first-generation protocols no less useful but it does mean that application developers will have to perform some custom validation; they can't count on the XML processor to do all validation for them.

Is there a better way? Let's trace the root of the problem. The reason we wanted to go with something like WDDX was that we wanted to process the information about people as application data. Trouble was, there were no ready-made tools that knew that the contents of `<name>` should be a string and the contents of `<age>` should be a number. So we used a WDDX encoding, in which we put the name inside a `<string>` element and the age inside a `<number>` element. We performed a transformation in which the type of the data became expressed in XML but the semantic meaning (or origin) of the data was lost. We gained the ability to process the information with any number of programming languages but lost the ability to declaratively validate that we are dealing with information about people.

Ideally, we want to find a way to use meaningful element names such as name and age that give us readability and declarative validation, yet at the same time attach a data type of "string" to name and "positive integer" to age. XML 1.0 doesn't offer this, but XML Schemas do. For example, the following schema will fit our needs:

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <xsd:element name="person" type="personType"/>
  <xsd:complexType name="personType">
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="age" type="xsd:positive-integer"/>
  </xsd:complexType>
</xsd:schema>
```

As you can see, XML Schemas are a significant improvement over DTDs. (For more information see Robert DuCharme's article "Replace DTDs – Why?" in *XML-J*, Vol. 1, issue 1.) First, they're expressed in XML, which makes their authoring and automated processing a lot easier. Second, they allow you to specify in much greater detail the structure of your XML. They also allow you to associate a data type – string, positive integer and many others – with element content. Last but not least, they let you specify new complex types (such as `personType` in the example above). Complex types are the key to combining data type information with validation information.

The work on XML Schemas is, of course, handled by the W3C. It's been a long and arduous process, primarily because XML Schemas try to do a lot more than what's shown above. In the beginning of April the sixth draft of the specification was released. Sources that are part of the working group say the end is near. We should hope so, because XML Schemas are of great importance to XML protocols.

## Second-Generation XML Protocols

Lacking support for namespaces and schemas, the architects of first-generation XML protocols kept reinventing the wheel. They all had to come up with ways to identify the XML messages used by the protocol, to serialize and deserialize data, and to manage other protocol aspects such as security and transactions. The end result was significant redundancy and repetition in most XML protocol specifications. Translation software had to be written to convert between different protocol formats.

This fragmentation raised the overall cost of adoption of XML protocols by businesses everywhere at a very inopportune time, just as B2B computing was really taking off. B2B relies on Internet application interoperability that's based on XML protocols. (For more information, see my article "XML for B2B" in the *XML-J*, Vol. 1, issue 2.)

Something had to be done – and it had to be done quickly. The only good solution to the fragmentation problem was the development of base XML protocol standards by an independent standards body such as W3C, IETF or OASIS. With namespaces and schemas, XML technologies had matured to the point at which it was conceivable that a single specification could address all relevant issues in an elegant yet extensible manner.

Around the turn of the millennium, there was a spike in activity. By that time, Microsoft had already published the Simple Object Access Protocol (SOAP) specification. SOAP 1.0 suggested a highly abstract and extensible mechanism for XML-based distributed computing. It opened the door for standardization of other protocol facets such as security and transaction management but didn't make any specific recommendations in these areas.

In January 2000 CommerceOne published an IETF draft on the requirements of XML messaging. The draft addressed issues of great importance to business messaging and thus the Electronic Business XML Initiative (ebXML) was formed. OASIS and the United Nations/CEFACT group manage ebXML. For more information read Bob Sutor's "Introducing ebXML" in the May-June issue (*XML-J*, Vol. 1, issue 2).

It was at this time that the W3C got involved. Right before XTech 2000 the W3C made an announcement that it was looking into starting an activity in this area: "We've been under pressure from many sources, including the advisory board, to address the threat of fragmentation of and investigate the exciting opportunities in the area of XML protocols. It makes sense to address this now because the technology is still early in its evolution and more resources for development will be available as XSL-FO and XML Schemas near completion."

At XTech 2000 there was a birds-of-a-feather (BoF) meeting on XML protocols that drew a crowd of XML VIPs. No answers were obtained but a lot of good questions were raised. Since then, Eric Prud'hommeaux of the W3C has been doing a lot of work analyzing the existing XML protocol space and organizing some of the activity on the W3C xml-dist-app mailing list.

In the beginning of May SOAP 1.1 was submitted as a Note to the W3C. The big surprise was that IBM coauthored the specification with Microsoft. Hopefully, this will limit the impact of politics on the standardization process and allow the W3C to fast-track work on XML protocols.

There is still uncertainty with regard to three important issues:

- How much do SOAP and ebXML have in common, i.e., should the two efforts coordinate?
- If there is an overlap in scope, who will coordinate and manage the joint process? The W3C, OASIS and UN/CEFACT all have their own political agendas and want to be associated with such an important development.
- How detailed should the specifications get? For example, should they go into the details of how certain types of security and transactions should be handled, or should they just put in place extensibility hooks and let someone else decide on this later?

At the 9th World Wide Web (WWW9) conference in Amsterdam, there was an "XML Protocol Shakedown" panel discussion in which some of these issues were addressed. Looking at how quickly standards bodies and vendors are willing to act, there's a good chance that by the end of the year we'll be close to having a solid specification for a second-generation XML protocol that will address the requirements of a broad set of distributed computing scenarios.

## What's Next?

This article has traced the evolution of XML protocols from simple first-generation solutions based on XML 1.0 to extensible second-generation

ation protocols such as SOAP. Namespaces and XML Schemas are the key enablers of this transition. There's a lot of industry activity around XML protocols and the W3C is close to starting a working group in this space.

While B2B is the most business-press-worthy topic of the year, XML protocols are clearly where the action is in the XML technology and standards space. Since ***XML-Journal*** is dedicated to keeping you informed of important developments in the field of XML, we'll be adding a regular column – "XML in Transit" – that will focus on XML protocols.

In the next issue I'll analyze the aftermath of WWW9 and the progress of SOAP through the W3C. I'll also address the Sun/Java standards play in the XML protocols space. If you'd like me to discuss some particular topic, drop me a note at [simeons@allaire.com](mailto:simeons@allaire.com). ☎

## XML Resources

**BizTalk:** BizTalk. Microsoft Corporation. See [www.biztalk.org](http://www.biztalk.org).

**Messaging Requirements:** "Requirements for XML Messaging." IETF Draft. See [www.ietf.org/internet-draft/draft-ietf-trade-xmlmsg-requirements-00.txt](http://www.ietf.org/internet-draft/draft-ietf-trade-xmlmsg-requirements-00.txt).

**Namespaces in XML:** "Namespaces in XML." W3C. (World Wide Web Consortium.) See [www.w3.org/TR/1999/REC-xml-names-19990114/](http://www.w3.org/TR/1999/REC-xml-names-19990114/). "NOTE Namespaces in XML" W3C. See [www.w3.org/TR/1998/NOTE-xml-names-0119.html](http://www.w3.org/TR/1998/NOTE-xml-names-0119.html).

**SOAP 1.0:** Simple Object Access Protocol (SOAP) 1.0. Microsoft Corporation. See <http://msdn.microsoft.com/xml/general/soapspec-v1.asp>.

**SOAP 1.1:** Simple Object Access Protocol (SOAP) 1.1. W3C (World Wide Web Consortium). See [www.w3.org/TR/2000/NOTE-SOAP-20000508](http://www.w3.org/TR/2000/NOTE-SOAP-20000508).

**WDDX:** Web Distributed Data Exchange (WDDX). Wddx.org. See [www.wddx.org](http://www.wddx.org).

**XML-RPC:** See [www.xmlrpc.com](http://www.xmlrpc.com).

**XMLSchema:Datatypes:** "XML Schema Part 2: Datatypes." W3C See [www.w3.org/TR/xmlschema-2/](http://www.w3.org/TR/xmlschema-2/).

**XMLSchema:Structures:** "XML Schema Part 1: Structures." W3C See [www.w3.org/TR/xmlschema-1/](http://www.w3.org/TR/xmlschema-1/).

### LISTING 1

```
<methodCall>
  <methodName>NumberToText</methodName>
  <params>
    <param>
      <value><i>4</i></value>
    </param>
  </params>
</methodCall>

The response could be:
<methodResponse>
  <params>
    <param>
      <value><string>twenty-eight</string></value>
    </param>
  </params>
</methodResponse>
```

### LISTING 2

```
<protocol:message>
  xmlns:protocol="http://www.protocol.s.org/ProtocolSchema">
    <protocol:headers>
      <trustMe:authInfo>
        xmlns:trustMe="http://www.trustme.com/SecuritySchema">
          <trustMe:userName>Joe Engineer</trustMe:userName>
          <trustMe:password>test123</trustMe:password>
        </trustMe:authInfo>
      </protocol:headers>
    <protocol:body>
      <!--Body goes here-->
    </protocol:body>
  </protocol:message>
```



## AUTHOR BIO

Simeon Simeonov is chief architect at Allaire Corporation.

[SIMEONS@ALLAIRE.COM](mailto:simeons@allaire.com)

# GCA

## www.gca.com





*Does current technology support the trend toward ubiquity?*

# XML: A Step Closer to Ubiquitous Computing

For several years the computing industry has discussed a state of total harmony among systems and software called *ubiquitous computing*. XML, a tremendous simplification of SMGL, is already competing to become the lingua franca of computing. However, the power of XML is found not so much in the direct benefits provided by the specification or related technologies, but in the indirect benefits measured in the larger sense (for instance, how do you describe the importance of ASCII?).

This article describes the puzzle of ubiquitous computing and how this new piece – XML – helps solve some of the puzzle.

## What Is Ubiquitous Computing and How Is It Affected by XML?

The latest incarnation of the ubiquitous computing concept is the new wave of business-to-business computing. In B2B computing, consumers and suppliers seamlessly complete shared transactions over a “virtual” common infrastructure and/or network.

If you agree that ubiquitous means “everywhere and all the time,” then

ubiquitous computing is clearly still only a concept. The question is: What does it take to interconnect computer equipment and have all applications seamlessly working together? The capabilities required for ubiquitous computing are:

1. *Connectivity*
2. *Application awareness*
3. *Application interaction*
4. *Context or semantic interpretation of calls to interfaces*
5. *Shared evolution*

Let’s see whether the current technology mix that supports each capability is mature or stable.

### CONNECTIVITY

Pick a medium: fiber, copper, wireless....Pick a configuration: LAN, WAN, MAN, Internet, intranet....Everything we need is already here, and more options are presented regularly. This capability, although advancing regularly, is stable and not affected by XML.

### APPLICATION AWARENESS

The first part to this capability involves applications that are designed to interact with one or more known peer applications or with higher- or lower-level ones. These are applications in which business requirements shape the functionality and system interaction. This is a very mature and stable domain.

The second aspect involves applications that are programmed to discover one another at runtime. This is often the realm of agent-based software. These

		Application Interaction Paradigm	
		Interactive (in general this requires open systems)	Batch
Application Awareness:	Interface found at design time	<ul style="list-style-type: none"> <li>• Must specify the format of the interface</li> <li>• XML can simplify formatting requirements</li> <li>• Maturity/stability – high</li> </ul>	<ul style="list-style-type: none"> <li>• Must be able to identify a file format</li> <li>• XML not required but can help with shared evolution</li> <li>• Maturity/stability – high</li> </ul>
	Interface found at runtime	<ul style="list-style-type: none"> <li>• Must provide a mechanism to “advertise and discover” the interface</li> <li>• XML can reduce the complexity of discovering “how to word the message”</li> <li>• Maturity/stability – low</li> </ul>	<ul style="list-style-type: none"> <li>• May not have existed previously</li> <li>• Made possible by self-describing nature of XML</li> <li>• Maturity/stability – low</li> </ul>

TABLE 1 How XML affects application awareness and interaction



# Software AG

[www.softwareag.com/tamino](http://www.softwareag.com/tamino)



# XML doesn't solve all the problems of computing but it does solve some. And it's important to identify those sections within the ubiquitous computing puzzle where XML does provide a solution



can be complex systems that, combined with an immature market, lead to an unstable capability. Emerging technologies and specifications such as JavaSoft's Jini and Microsoft's Universal Plug and Play aim at enabling these dynamic networks of cooperative devices and programs.

## APPLICATION INTERACTION

A rudimentary distinction between interaction paradigms is whether a system is "batch" or "interactive." (For clarity, a *batch* interface is one in which one program writes a file as output, and a subsequent program reads the same file as input. An *interactive* interface is one in which one program invokes an API on another program. The word *program* is used loosely here.) Defining how XML affects application awareness and application interaction can be very convoluted. A simplified version of the problem is shown in Table 1.

## CONTEXT INTERPRETATION

True communication means that the origin and destination of the message understands the semantics of the message. Regardless of whether or not applications have the capability to interact simply because they use common types at the interface level, the applications must be able to understand the context of the communication. This applies to all cells in Table 1. Without the ability to understand context, ubiquitous computing can't happen – possibly the greatest challenge to full-scale implementation. The maturity and stability in this capability are low.

## SHARED EVOLUTION

A rarely considered phenomenon regarding new technologies, shared evolution is clearly the harshest reality for system integration and deployment in the real world. No matter how nicely you try to spin this (e.g., "We can leave the old API intact until all applications

migrate to the newer version, then we'll remove the old one"), the fact is that sometimes there is no alternative but to punt, and several applications must be redeployed at the same time. This is a problem that has a medium maturity but a low stability since the solution is relatively new.

## What Solutions Does XML Provide?

XML doesn't solve all the problems of computing but it does solve some. And it's important to identify those sections within the ubiquitous computing puzzle where XML does provide a solution. XML will improve the maturity and stability of the capabilities that are required to reach ubiquitous computing in the following areas.

## XML AND APPLICATION INTERACTION

While CORBA and IDL provided a language- and platform-independent way of defining system interfaces, they're very strict with respect to types and ordering. Changes to a CORBA message result in changes to the IDL, which results in changes to all systems using that IDL. With XML this problem can be solved. Messages can be passed as strings, with XML-compliant data streamed into that string, allowing additions and certain types of changes to arguments without breaking the existing interfaces. The XML strings need only adhere to a common form, as described in a DTD or schema.

Further, XSL is capable of transforming XML into other XML. XSL may therefore provide a bridge between systems that don't adhere to the exact same XML structure but have a commonality in the content. This will help with information exchange between two systems that originally weren't intended to interface. In this way XML avoids, to some extent, the need for collaborators to adhere to the exact same interface. This is an important

capability in an environment in which being adaptive is far more important than being optimized. Compare the present state of the industry to the brittle world of electronic data interchange (EDI) of the last decade.

Organizations within several industries are already exchanging or attempting to exchange data with other companies as a part of normal business: telecommunications companies exchange billing and call records, hospitals and doctors exchange healthcare information, and so on.

## XML AND SHARED EVOLUTION


The most notable impact XML can have on shared evolution is the elimination of fixed record formats. Information streamed to a file based on the XML standard can routinely change unbeknownst to the recipients of the file. Those recipients use a parser to get the data they need and don't really care about the size of the file or the field order. This will allow one system to deploy at one point, including changes to the file layout, and other dependent/downstream systems can deploy later.

With the flexible extensibility of XML, this could be one of its greatest benefits.

Now data requirements from various organizations can evolve somewhat independently since upstream systems can add new fields to a file without affecting downstream systems. Later, the downstream systems can be enhanced to make use of the new data.

XML can affect the client portion of an application. Specifically, it lets the developer use the same client configuration information for several different versions of the client. Notably, those versions can vary by device type. In other words, the same XML can be used in several different ways, depending on the corresponding use of XSL. Exactly how much this will affect us in the long run has yet to be determined, but development shops are already hard at work discovering the innovative uses.

## Conclusion

Now is the time for your company to determine the benefits available to it through selective use of XML. Many early adopters are already taking advantage of XML in their systems. If you're not sure where XML would benefit you the most, keep your ear to the ground for the early success stories. 

## AUTHOR BIO

John King, a delivery director for The Technical Resource Connection, Inc., is a systems architect and manager. He's currently directing two projects (in telecommunications and supply chain management) using an architecture-driven approach and advanced development techniques.

 JOHN.KING@TRCINC.COM

# Geek Cruises

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



*Remote procedure calls provide easy access to distributed resources*

# XML-RPCs and Java

Computer paradigms are cyclical – they come, go... and come back around. Remote procedure calls (RPCs) are one such paradigm. This month's column focuses on the promise of ubiquitous communications in the Internet using XML-RPCs and RMI.

XML-RPCs are intended to provide a language-neutral remote procedure mechanism that works over the Internet. However, before getting started, here's some background.

## A Little Bit of History...

RPCs were introduced with UNIX systems. However, RPC code wasn't portable across different UNIX operating systems. The purpose of RPCs was to allow programmers to develop network applications without requiring them to be socket programmers. RPCs provided an interface definition language capable of generating communication stubs and skeletons for multiple programming languages. This was accomplished by using a compiler program on top of the specified IDL.

Later on, a consortium was established to develop an RPC mechanism – called DCE RPCs – that could function across operating systems. These provided a compiler and libraries whose stubs could be compiled and executed on multiple operating systems, allowing clients and servers to produce code that was portable across heterogeneous operating systems. Developers needed only to com-

pile the code on the new platform and the application would work. This was good, but DCE wasn't intrinsically designed to deal with object-oriented languages.

The OMG (Object Management Group) defined an object request broker (ORB) specification called CORBA (Common Object Request Broker Architecture) to provide an object-oriented mechanism for RPCs. Like RPCs, CORBA provides an IDL used by developers to define object interfaces for multiple languages. However, the stubs and skeletons developed by CORBA IDL compilers aren't necessarily portable across multiple ORB vendors. Communications are standardized between heterogeneous CORBA servers through the use of a communication protocol called IIOP (Internet Inter-Orb Protocol).

## RMI Is Java's RPC

The Java platform provides its own object-oriented RPC mechanism called remote method invocation, or RMI. This provides a native Java mechanism that allows objects to communicate in memory across multiple virtual machines. This communication allows local objects to invoke methods on remote objects as if they were located on the same process space. RMI uses RMI URLs to locate a remote object on another virtual machine. Using Java's object serialization, an RMI object is capable of being transported between virtual machines and accessed in its totality by the distributed process. Object serialization in RMI is one of the main differences between RMI and CORBA. Another difference is that CORBA provides a cross-platform, cross-language architecture, while RMI was designed to solve the problem of distributed object communications in the Java world. The last interesting point about RMI is that it can be tunneled over HTTP.

## RPCs and XML

Enough history! What about XML-RPCs? XML-RPCs are remote procedure calls executed over the Internet. What this means is that they leverage the HTTP protocol to execute distributed processes that live on the network or on the Internet. It also means that XML-RPCs are invoked against a special-purpose HTTP server (see Figure 1).

XML-RPC requests are sent using an HTTP POST. An HTTP POST allows a client to send large quantities of data between the application and the Web server. The body of the HTTP request is defined using XML and the response to the request is returned in XML. The first part of the XML HTTP request contains an optional URI for the POST, a required User-Agent and Host name that needs to be identified, a text/xml Content-Type and a correct Content-length for the request. The URI, which helps route the request to the call that handles it, can be empty if the server only handles XML-RPC requests. An example of the first part of the request follows:

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: rpcxml.experiments.com
Content-Type: text/xml
Content-length: 199
```

In this example all requests are routed to the RPC2 responder on host `betty.userland.com` running a Frontier Web server with a content type of text/xml and a length of 199 characters.

The second part of the XML HTTP request is made up of a single XML `<methodCall>` structure. Inside the `<methodCall>` a `<methodName>` tag needs to be identified with the name of the method being called. The name can be used to represent a script file responsible for answering the request or a file in the document hierarchy.

## 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.

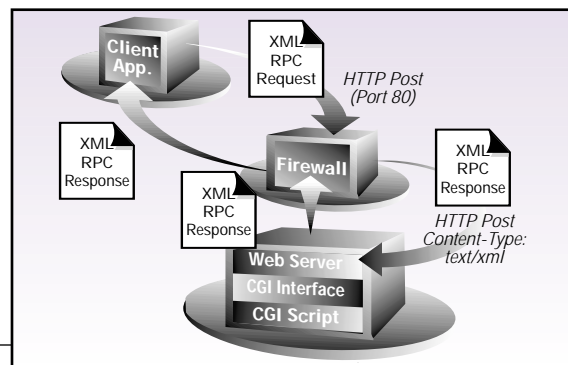


FIGURE 1 XML-RPC execution framework

Parameters are represented using a `<params>` tag inside the `<methodCall>` tag. A singular `<param>` tag is used inside the `<params>` tag to identify each of the individual parameters to be passed to the procedure. Each of the `<param>` tags contains a `<value>` tag. The `<value>` tag contains a description of the parameter type and its value. An example of the second part of the request follows:

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

  <methodName>experiments.getDNSName
</methodName>
  <params>
    <param>

      <value><string>129.107.10.3</string>
</value>
    </param>
  </params>
</methodCall>
```

In this example the method name represents an object called "experiments" with a method called "getDNSName". This procedure has one parameter of type "string" and a value of "129.107.10.3". Remember that the return type of the procedure will be an XML structure.

The format of the response is divided into an HTTP header response and the XML structure. The XML structure contains a `<methodResponse>` tag that contains a `<params>` tag. The `<params>` tag contains a single `<param>` tag, which in turn contains a single `<value>` tag. The `<value>` tag contains a tag that describes the type of the result being returned and the value of the result.

The body of the response is a single XML structure, a `<methodResponse>`, that contains a single `<params>` tag. The `<params>` tag contains a single `<param>` tag. The `<param>` tag contains a single `<value>`. An example of the XML structure response follows:

```
<?xml version="1.0"?>
<methodResponse>
  <params>
    <param>
      <value><string>pokemon.experiment.com
</string></value>
    </param>
  </params>
</methodResponse>
```

In this example the method response contains one string result with the value "pokemon.experiment.com". Responses can also return fault structures with a fault code consisting of a fault code ID and a fault description string.

This protocol was designed to leverage the HTTP protocol across firewalls. Firewalls can filter HTTP POST requests with text/xml content-types. The only Web server capability expected by the protocol is CGI support. The programming audience of the protocol is HTML developers. One advantage of XML-RPC over RMI is that the client developer isn't required to write Java code, only to create an XML structure to be posted and parse the returning XML structures.

## XML-RPCs and Java

XML-RPCs and Java can be viewed as complementary technologies. Java's RMI technology adds the power of distributed computing to the mix. A Java servlet is created to handle the XML-RPC POST requests. The `<methodName>` defined inside the `<methodCall>` tag contains the RMI URL appended with the method name being called. This information is placed inside the `<methodName>` tag. An example of the method name follows:

```
<methodName>rmi://rpcxml.exp.com/networkinfo/getDNSName</methodName>
```

In this example the RMI URL defines a remote object called networkinfo on host rpcxml.exp.com. The method name to be executed on the remote object is getDNSName. The parameters and values remain the same as in the previous example.

The servlet is responsible for retrieving the RMI URL from the `<methodName>` tag and looking up the object on the RMI registry. Once the object is retrieved, the method to be called is accessed by the servlet using Java's reflection. The method's parameters can be constructed from the `<params>` information contained in the XML structure and its execution takes place using the reflection API. The returned object is decomposed into an XML structure and returned to the client.

The servlet is responsible for parsing the input XML structures and generating the output XML structures. It's also responsible for mapping XML-RPC data types into Java objects and vice versa. Table 1 is a list of suggested XML-RPC Java mappings from the UserLand group ([www.userland.com](http://www.userland.com)).

If the servlet isn't capable of resolving the XML `<param>``<value>`, mapping it will return a fault to the client application. Faults are handled by the servlet by capturing Java exceptions and generating XML-RPC fault structures. These structures need to contain a numeric value assigned to the exception type and the textual description associated with the exception.

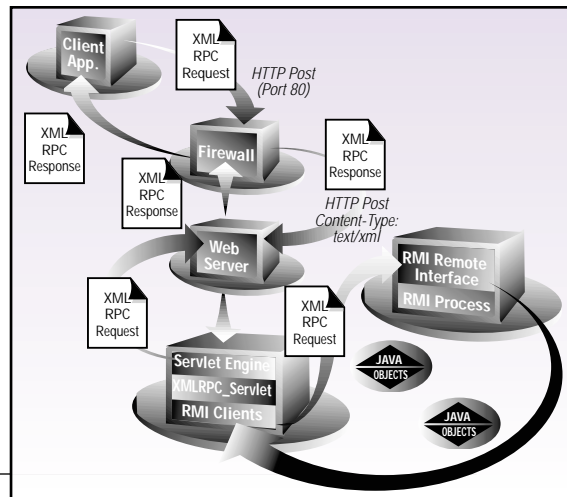


FIGURE 2 Execution framework for RMI/XML-RPC invocations

XML-RPC DATA TYPE	DATA TYPES GENERATED BY THE PARSER	TYPES EXPECTED BY INVOKER AS INPUT PARAMETERS TO THE RMI OBJECTS
<i4> or <int>	java.lang.Integer	int
<boolean>	java.lang.Boolean	boolean
<string>	java.lang.String	java.lang.String
<double>	java.lang.Double	double
<dateTime.iso8601>	java.util.Date	java.util.Date
<struct>	java.util.Hashtable	java.util.Hashtable
<array>	java.util.Vector	java.util.Vector
<base64>	byte[]	byte[]

TABLE 1 XML-RPC Java mappings

Figure 2 describes the relationship between the Web server, servlets, RMI objects and the XML-RPC request.

One of the goals of XML-RPC is to provide easy access to distributed resources; the Java abstraction provided by the XML-RPC protocol supports this concept. The combination of these technologies allows non-Java client applications such as HTML pages to access distributed RMI objects, which in turn allows HTML developers to use RMI process without having to code in Java. Another advantage of this marriage is that Java applications are portable across multiple operating systems, making them perfect for developing distributed object applications.

## Summary

As we've seen, RPCs have evolved a great deal since their inception. As companies extend their boundaries into the Internet space, the combination of technologies like servlets, RMI, CORBA and XML-RPCs will become more critical to access services over the Web.





*Your mainframe on your cell phone*

# XML-Based Enterprise Information Portals

**E**-business, B2B, enterprise information portals (EIPs) and XML are the leading buzzwords of our industry because information – and its efficient management – is at the heart of any e-business environment. XML is the standard for the markup of information in Web-based Internet/intranet and extranet applications and this article provides future users of XML with a blueprint of how to make the best use of these new and converging technologies.

This article will start by looking at how enterprises can use XML to connect different systems throughout their organizations, then go on to show how they can use portal technology to serve the needs of internal and external users. Finally, I'll show how all of this provides an e-business-enabling layer for commerce-oriented applications.

## The XML Backbone

### LEGACY SYSTEMS

At the simplest level EIPs provide an efficient means of managing secure and personalized access to corporate infor-

mation resources. However, one of the problems we had to face very early on was that this alone wasn't enough.

All too often, the resource an EIP is supposed to manage isn't ready to be accessed, especially via the Web. One of the first tasks while building an EIP in a large-scale enterprise is to arrange for the Web-enablement of very diverse data sources.

Clearly, XML is the key candidate for these categories of problems. Instead of using the proprietary data formats of the data sources, XML is a natural fit as it describes the information handled in a standardized way.

Moreover, in XML we also get the ability to work with stylesheets to provide various renditions of our data. And once extracted into XML format, not only can EIPs use the data set but so can other applications, especially if they're already XML-enabled.

## REUSABLE INFRASTRUCTURE

According to a recent GartnerGroup report, a typical enterprise will devote 35–40% of its programming budget to developing and maintaining "extract and update" programs whose sole purpose is to transfer information between different databases and legacy systems (see Figure 1).

While in the old pre-XML days dozens of programmers had to waste their talents on repeatedly developing

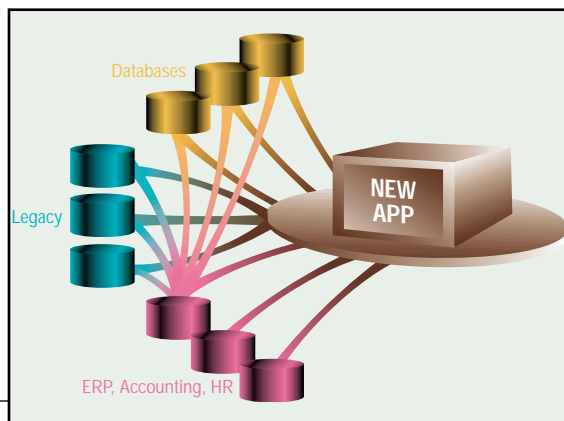


FIGURE 1 Before XML – little reusable infrastructure

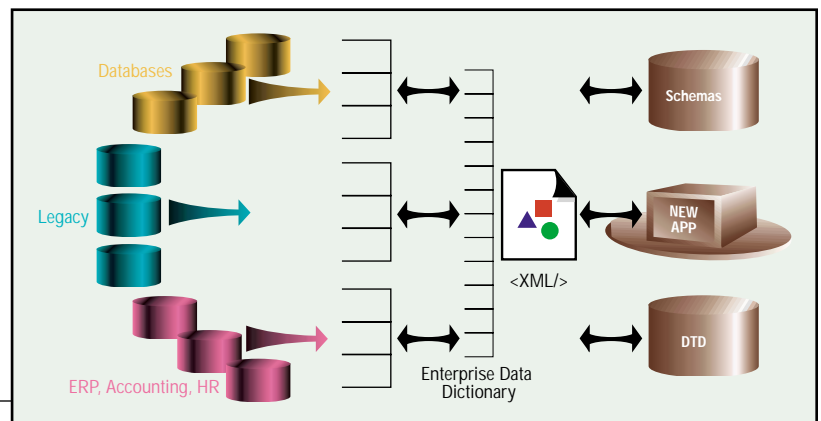


FIGURE 2 XML-based infrastructure - the XML backbone

# Cape Clear

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

extract and update programs, today – using XML – we can slowly start to develop an abstraction layer in which all the information in an organization is being passed around in XML (see Figure 2).

As a matter of fact, the XML backbone per se is an abstract concept, a way of thinking. It makes no assumptions about underlying transport protocols. Enterprises can implement an XML backbone using whatever technology/protocol (or rather mix thereof) they want, whatever fits into their existing and planned IT infrastructure.

Besides the obvious integration of back-end systems, the integration of other XML components more oriented toward documents – such as document management, editors, link managers and so on – is another level of the XML backbone that's also of strategic importance.

### IT'S NOT A SHRINK-WRAP WORLD

A word of caution at this point: all that's been said so far about the XML backbone might sound too good to be true. Well, you can get there, but it will require a significant commitment to a strategic initiative...and it will certainly take time and money.

It's not a shrink-wrap world, except for well-defined subproblems. Don't expect a single vendor to come to you with the panacea for all your IT problems in five minutes. These are difficult problems that skilled and experienced IT experts have been trying to solve for many years; XML is merely a new way of looking at it. Granted, as with any other technology, you'll be able to achieve partial success in a relatively short period of time. But the truly complex problems typically take a long(er) time to solve.

One of the more promising developments in this area is that more and more vendors have started shipping XML-capable releases of their software. Many others have recognized the need to provide connector technology for integrating various systems into the brave new world of XML.

### An XML-Based EIP

What do we really mean when we say an "XML-based EIP"? Typically it will include one or more of the following aspects.

#### XML FOR METADATA

Even though we might wish all the data in the world were already in XML, we're still far away from that – very far, as a matter of fact. For the time being, the harsh reality is that EIPs will have to deal

with data that doesn't come in XML format, such as word-processed documents, spreadsheets and graphics. XML still comes in very handy in these cases as a wrapper for such BLOB objects (see Figure 3). In this way, such metadata can be handled very efficiently.

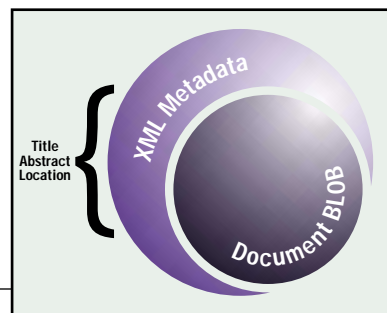


FIGURE 3 Using XML to wrap metadata

#### XML FOR DATA

The use of XML in data distribution relates back to the foregoing section about the XML backbone. Through a portal, we pass XML information to end users (rendering it with some sort of stylesheet mechanism) or to some other system for further processing. From a portal's perspective it doesn't really matter whether your content ends up at the user's desktop or some other machine.

If you're so fortunate that your portal vendor provides some fancy connectors in the back end, combined maybe with the ability to insert your own business

rules using scripting or other means, you can do even more interesting things, such as write back to data sources or create complex applications driven by these scripts.

One example of what you can do with XML on the server side, especially combined with a persistent DOM on the server, is the XPages environment (see Figure 4). XPages was developed by DataChannel and has been submitted to the Apache XML open-source project.

### XML-Based Software Development

I call this "eat your own dog food." Wouldn't it be silly for a vendor to tout XML whenever and wherever possible if he or she doesn't use XML for the software itself? Indeed, as it turns out, XML can be of great use in the software development process. Without getting specific about our own software, I would like to share some of the results of our R&D at DataChannel, since it may be applicable to your own software development.

I've mentioned earlier the XML backbone as an enterprise-wide initiative. During our product development, we also created a product-internal XML backbone – an abstraction layer for all internal data sources to be abstracted into a single DOM-interfaced data structure. All the application developer gets to see is the DOM, although in the background we interact with filesystems, RDBMS and LDAP/ADSI components.

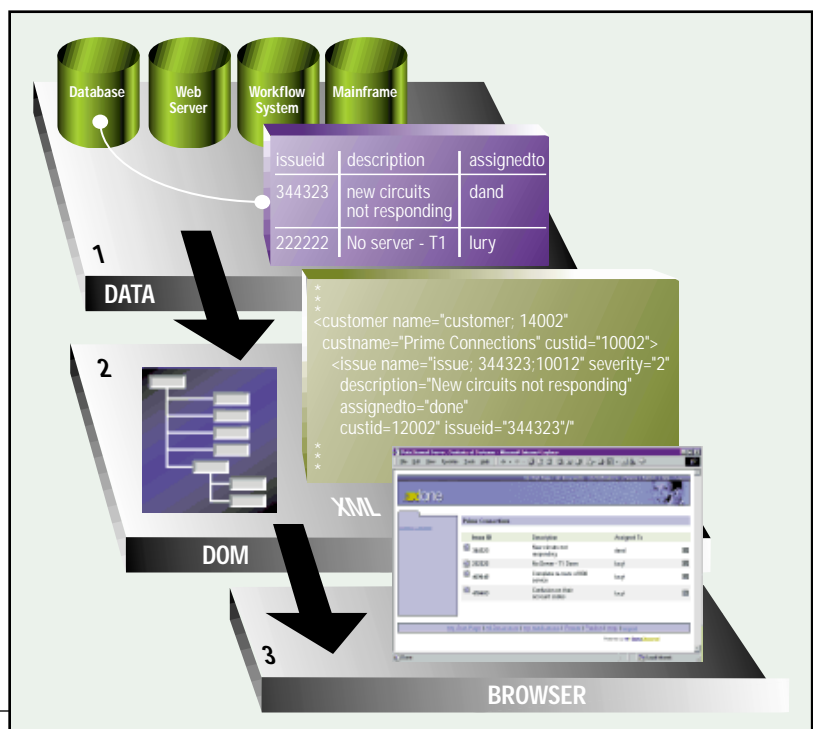


FIGURE 4 XML delivery, from the back end to the front end

# SD EXPO

[www.sdexpo.net](http://www.sdexpo.net)

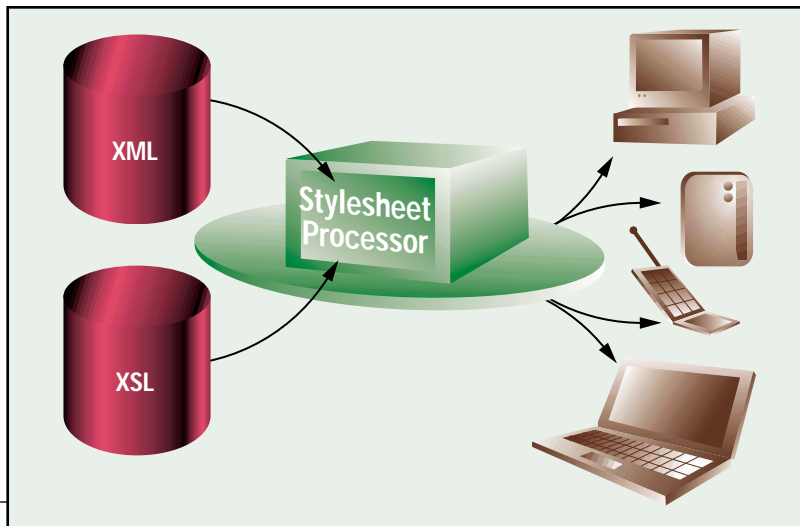


FIGURE 5 Create once – publish in different ways

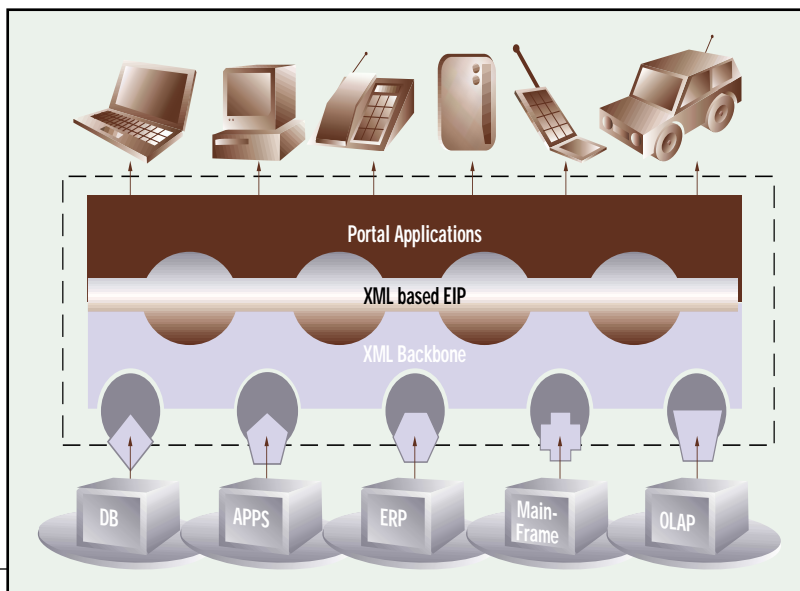


FIGURE 6 A completely XML-based EIP scenario

## AUTHOR BIO

Norbert Mikula developed one of the first XML parsers, NXP, and has been engaged in XML-related efforts since the early days of this standard. As CTO, he is now responsible for directing DataChannel's efforts in the XML standards bodies and its strategic product direction. He also serves as CTO on the board of directors of OASIS.

Following this approach, we were also able to separate content from display, and could even separate the programming logic. Using XSL/CSS, therefore, we have full control over the look and feel of the portal, and because we believe this is of great value to the entire XML community, the core component of this module has been submitted to Apache.org.

## Business to Anywhere

The browser, whether laptop or desktop, is losing its position as the only way to access information from the Internet. As our society becomes increasingly mobile, our information systems design must follow the trends.

The GartnerGroup estimates that by the end of 2005, more than one billion

mobile phones will be in use worldwide. Cellular phones and PDAs are just the beginning of a whole new era of information mobility, and XML is destined to be one of the leading enabling technologies for this new technology wave.

## Reuse

Content already marked up in XML can be reused for any output data format needed (see Figure 5). The same XML content can be rendered for output to a browser (HTML), to a cell phone (WML or HDML), to a PDA (maybe a subset of HTML) or any other conceivable data format. Granted, this doesn't yet solve the problem of particular transport protocols to a device but, at least from a data perspective, everything is in place.

## DEVICE MARKUP LANGUAGES

XML provides very flexible means for designers of output devices to create data formats that are most appropriate for the requirements, including the constraints of a particular device. XML is particularly suitable for this, as it's a metalanguage for creating markup languages. Wireless markup language (WML) is just one example.

## A Not-So-Traditional B2B Landscape

When you hear the term *B2B communication*, what comes to one's mind all too often is "machine-to-machine" integration. In other words, how do I enable the exchange of data between my own enterprise systems and those of my business partners, such as suppliers and customers? Some of the systems offered in today's market will even allow you to manage the exact steps for the exchange of business data, and to monitor the execution of these business processes, via a graphical user interface.

However, this very narrow view of B2B leaves out two important aspects:

1. **Document-content-oriented B2B:** Exchange of design drawings, contract negotiations, RFQ, RFP and RFI development, and so on. These types of operations are typically conducted between human participants and involve collaborative work on unstructured content. XML transactions alone are sufficient when only commodities are involved, but aren't sufficient for more complex business processes between companies.
2. **Human decision-making:** Typical B2B processes involve more than the exchange of transactions between machines. Usually, human beings will be involved and have to make quick and informed decisions that allow a business process to continue smoothly to the next step. What's the point of high volume transactions between machines if the human being(s) involved take hours/days to do their part?

Enterprise information portals, using XML technologies, are well positioned to cover these two areas. Both from a content and a metacontent perspective, XML provides the necessary power.

To conclude, Figure 6 shows you how it all fits together: from the XML backbone through the portal to any device, anywhere. ☺

NORBERT@DATACHANNEL.COM



# Simplex Knowledge Company

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

# XML STANDARDS FOR CUSTOMER INFORMATION QUALITY MANAGEMENT

To leverage  
your strategic  
information  
assets you need  
to handle your  
metadata  
correctly

**T**he three key factors that contribute to the way we do our business today are corporate globalization and internationalization, an increase in the number of company acquisitions and a rapidly changing and increasingly competitive business environment. As a result, organizations recognize the immediate and urgent need to leverage their information assets in new and more efficient ways.

At the heart of these information assets is *enterprise data*, the data collected during the normal course of business. This is regularly aggregated, combined and analyzed to provide the information needed for corporate decision-making. Once viewed as operational or tactical in nature, enterprise data is now used for strategic decision-making at every business level. Managing the strategic information assets and providing timely, accurate and global access to enterprise data in a secure, manageable and cost-efficient environment is becoming critical.

Metadata – the information an enterprise stores about its data – has become the critical enabler for managing the integrated information assets of an enterprise. It's also, however, the weakest link in the information management chain.

The proliferation of proprietary data management and manipulation tools has resulted in a host of incompatible information technology products, each processing metadata differently. End users suffer from inaccessible and incompatible metadata locked into individual tools. Metadata has thus become the number one integration problem in the area of enterprise information and data warehouse management.

For enterprise-wide information management you need global and efficient access to shared metadata by all the heterogeneous products found in today's information technology environment. To use tools efficiently, users must be capable of moving metadata between tools and repositories.

Such tools are often provided by different vendors and run on different hardware and software platforms. To integrate the different tools and repositories in an enterprise environment, they must all share metadata in the same way, which means that it must be stored, managed and interpreted in a consistent way by all users. This is a complex problem, often considered too difficult to solve. But unless it's formally addressed, most corporate information management initiatives will be under severe stress.

## Customer Information Quality

Research shows that it's much more cost-effective to retain and invest in your existing customer base than to build or buy market share. One customer lost through ineffective marketing means a massive expenditure on acquiring new customers to make up the lost revenue. This is more apparent as we move into the e-business environment.

As companies move to establish more effective relationships with their customers, the need to achieve a complete view of each customer's dealings is recognized as critical. At the same time the cost and quality of interacting with customers is now a major business issue.

A direct result of this is the development of company-wide customer relationship management strategies, representing a combination of business processes, information management tools and, importantly, customer data. Customer data forms the foundation to build effective customer relationships. To be effective, customer data must meet the highest possible standards of both quality and integrity.

Often the impact of poor-quality customer information is fully understood only when attempting to unify customer data from disparate business systems. While data within individual databases may be fit for the purpose for which it's collected and used, combining data from a range of sources for a new and different purpose poses a real threat to the effectiveness of entire customer relationship initiatives.

Many organizations – in an effort to compete in a customer-centric world and move away from the inflexible, product-based approaches – are investing heavily in building customer information systems such as Data Warehouses, Data Marts, CRM (Customer Relationship Management) systems, Operational Data Stores and Single Customer Views. When an organization recognizes the need for a major customer initiative, they often ignore, due to time or budget restraints, the single most important component that can change the organization's bottom line: the quality of the customer information. Every organization has the answer within their grasp, but they get caught up with building new systems or trying to improve existing processes and systems.

Many CRM systems or data warehouses start with data migrations or conversions, involving movement of data, which raises many questions such as:

- What information do we have?
- What's the data that goes into this information?
- What information should be moved?
- What information can be moved?
- What's the quality of this information?
- How much does this poor-quality information cost the business?

Often organizations can't answer all these questions without the assistance of external experts who can develop a business impact assessment to show which data has priority for a particular business purpose and outline a data management strategy to maximize the strategic value of the data.

The bottom line is that reliable and accurate customer information is now more essential than ever in establishing effective customer relationships, and therefore customer information quality management is critical.

## E-Business and Information Quality

The Internet threatens to turn customer marketing on its head. While it has created tremendous opportunities for relating to customers on a one-to-one basis, it has also created a nightmarish challenge for companies struggling to understand this new way of interacting with customers.

Despite the tantalizing opportunity it represents, e-business is still an inefficient way to attract and retain customers. Customer acquisition costs on the Internet have skyrocketed to US\$65–250 per customer. Churn is up because more than 50% of Internet companies can't respond to their customers. Despite its promise, companies don't know and can't relate to their customers via the Internet due to the poor-quality customer data they maintain. To make matters worse, Internet customers are a global audience. Global customers require 24-hour-a-day, seven-day-a-week service, have language differences and multiple data and shipping formats – and vast differences in demographics, tastes, preferences and so on.

The general belief is that the more we know and understand about each customer the higher the chances they will purchase and return. But in reality companies have always been challenged by the wide-ranging, disjointed, duplicate data they collect and store. Customer service representatives rarely know what's in the billing system, and marketing almost never has a consistent, clear view of a customer's profile. The Internet exponentially complicates these challenges. With the Internet there's no one present to validate data as it is entered into the system, with the result that there's a huge potential for information and business errors, which in turn leads to poor marketing results and misdeliveries of products.

Retaining e-business customers by maintaining reliable and accurate information about them is therefore the most critical current problem in an e-business environment. And customer-information-quality management is the answer to this problem.

## XML for Standardizing Data

We could be at the beginning of a new era in data interchange – i.e., the exchange of information/enterprise data between programs and systems. In the not too distant future we may look back in amazement on the islands of data we have now and wonder how we could possibly have operated under such limitations. The standard that will allow data to flow as freely as e-mail already does today is XML. It may have many uses, but one of the most useful and powerful is its ability to build data interchange formats using metadata information.

XML is now being widely adopted and is clearly becoming an industry standard for data interchange and for delivering content on the Internet. For business use its most important advantages are:

- It provides an open, nonbinary, platform-independent language for data interchange.
- It makes the internal structure of documents/information evident to machines.
- It renders the documents/information much more reusable.
- It classifies information for easy search and retrieval.

These advantages have won XML a rousing welcome beyond the Internet community, and this century will belong to XML. The increasing support for XML as the de facto standard for data and metadata management is due to the fact that anything to do with data/information content will benefit from XML technologies.

The success and growth of XML as a data standard can be measured by the number of XML Document Type Definitions that have been defined and published for specific business data interchange. A DTD defines the grammar for a markup language. It accounts for the *extensible* in eXtensible Markup Language, and is how you define a new markup language (often called a *dialect* of XML).

A DTD grammar defines the following rules:

1. What elements may exist
2. What attributes elements may have
3. What elements may or must be found in other elements and in what order

As XML grows, it's likely there'll be an ever-increasing number of popular DTDs that we'll be able to download and use without going to the trouble of writing our own. To date there are more than 100 DTDs defined for various XML applications and the number is growing every day, including:

- Chemical Markup Language (CML)
- Voice Markup Language (VoxML)
- Process Interchange Format XML (PIF-XML)
- Ontology Markup Language (OML)
- Wireless Markup Language (WML)
- Bipolymer Markup Language (BIOML)
- Theological Markup Language (ThML)
- Java Speech Markup Language (JSML)
- Telecommunication Interchange Markup (TIM)

- Online Trading Protocol (OTP)
- XML Electronic Data Interchange (XML/EDI)
- XML Interchange (XMI)
- ColdFusion Markup Language (CFML)
- Weather Observation Definition Format (OMF)
- Mathematical Markup Language (MathML)
- Precision Graphics Markup Language (PGML)
- Astronomical Instrument Markup Language (AIML)
- OpenFinancial Exchange (OFX)
- Human Resources Markup Language (HRML)
- Synchronized Multimedia Integration Language (SMIL)

In recent times two technologies have dominated the Internet: XML and Java. In the words of Jon Bosak, known as the father of XML technology (and chairman of the XML Working Group of the World Wide Web Consortium), XML and Java technology are “the Yin and Yang of cross-platform technology.” XML gives you data portability and Java gives you software code portability: they're a perfect fit.

In a recent survey by the Cutter Consortium, 46% of XML users claim that 76–100% of their XML projects are successful – suggesting that XML's current user base is made up of pioneers and early adopters who are technology enthusiasts.

## XML Standards for Customer Information Quality

XML plays a significant role in the efforts many companies are undertaking to integrate e-business and CRM applications with their enterprise systems. The data to support real-time e-business is contained in legacy, back-office systems. Comprehensive CRM solutions are required to access data in a variety of disparate systems to achieve a complete picture of the customer relationships. As previously discussed, customer-information-quality practices are essential to the success of such CRM initiatives.

XML's usefulness applies here: companies can define XML grammars that deal with customer relationship management. However, multiple, vendor-proprietary XML grammars for customer relationship management make things more complicated when data is exchanged across product sets as special grammar converters are needed to move data between product sets. The best solution would be to develop an open, vendor-neutral, industry-standard XML grammar (DTD) for describing customers. This standard would assist in managing customer information quality.

## Name and Address Markup Language

Customer data consists of many components but the key identifier of a person or company is their name and address.

As a data type, name and address is very difficult to manage. The data is often volatile – customers come and go, addresses change, names change. The data is often cluttered when entered. Name and address fields on data entry screens are usually free-format and users sometimes enter comments without any edits. Name and address is subjective...it can be written in a number of different ways and still be the same. There's no application-independent standard to represent name and address data and measure its quality. This problem is further compounded in a global market by the different ethnic backgrounds of name and address data.

---

“ XML gives you data portability  
and Java gives you software code  
portability: they're a perfect fit ”

---

# XML Bootcamp



There are, however, a number of name and address standards available throughout the world. To a large extent they've been designed with a particular business requirement in mind – for example, the expedient delivery of a piece of mail. While that particular standard might be appropriate for the specific purpose for which it was designed, frequently it's unsuitable for other purposes.

One of the business units of the company of which I am technology manager – Cognito, Inc. – is called MasterSoft International. MasterSoft has over a decade of extensive experience in the area of customer-information-quality management, specializing in name and address data management in business areas such as telecommunications, insurance, finance and government – experience that has revealed the lack of an adequate standard for customer information quality and the importance of such a standard in data quality management, database design and EDI.

With the advent of XML as a de facto standard for representing data, MasterSoft has developed an application-independent XML standard for name and address data management called Name and Address Markup Language (NAML). NAML doesn't include all the address components throughout the world – but that's where the power of XML comes into play. It's extensively scalable and extendable, allowing NAML to evolve as additional components are identified.

## Name and Address Markup Language

Let me explain NAML DTD in more detail by looking at its two component parts, the name section and the address section.

### NAME SECTION

The name section of the NAML DTD defines various ways of marking up name data. If the entity is a company, then the <company> tag can be used. If the entity is a person, then either the <person> tag or the <person\_details> tag can be used. The whole name of the person must be enclosed within the <person> tag; however, the <person\_details> tag allows the name of the person to be broken down to the following:

```
<!ELEMENT person_detail s
  (title*, first_name?,
  middle_name*, last_name?, suffix*)>
```

The occurrence indicator \* allows that tag to appear more than once in the XML document. Hence the NAML DTD allows for person entities with multiple titles, middle names and suffixes. An example XML file that would be valid according to the NAML DTD is shown in Listing 1.

### ADDRESS SECTION

The address section of the NAML DTD defines various ways of marking up address data. There are three tags that can be used:

#### 1. <address>

Contains no subelements. If it's used, all the address data pertaining to an entity must be enclosed in this tag.

#### 2. <address\_lines>

Contains the subelements <address\_line1>, <address\_line2>, <address\_line3>, <address\_line4>, <address\_line5> to allow the address data pertaining to an entity to be split up into multiple address lines.

#### 3. <address\_details>

Contains many subelements relating to postal information, street information, place information, locality information, state information, postcode information and country information. This is to allow the address data of an entity to be broken down into the specific elements of an address.

All three address tags have the common attribute "address\_type". This is a required attribute to allow for customer entities that have mul-

iple types of addresses, e.g., billing address, residential address, postal address and so on. A valid XML file using the above tags and attributes is shown in Listing 2.

The example in Listing 3 shows how an address can be further broken down to its elements.

To date, MasterSoft has defined over 75 XML tags in NAML to represent name and address elements. This is continuously evolving as new name and address structures are encountered.

## Customer Identity Markup Language

Although name and address data is the key identifier of a customer, other data can help to uniquely identify a customer. Customer addresses frequently change, and it isn't feasible to link the customer across multiple addresses with just name information.

Compare, for example:

Ram Kumar  
2 Virgo Place  
Erskine Park, NSW 2759  
Australia  
Mobile: 0412-758025  
rkumar@hotmail.com

and

R. Kumar  
MasterSoft  
Level 12, 67 Albert Avenue  
Chatswood, NSW 2067  
Australia  
Mobile: 0412-758025  
rkumar@msi.com.au  
rkumar@hotmail.com

As this shows, it's nearly impossible to uniquely identify the customer with the name alone. Customer-centric data such as telephone numbers, e-mail addresses, account numbers and credit card numbers are necessary before you can achieve this.

It's therefore important to define a standard for representing all forms of customer data that constitute the basis of customer-information-quality management. MasterSoft has defined an application-independent XML standard for customer information quality called Customer Identity Markup Language (CIML). CIML provides a framework for representing different data about a customer and thus helps to uniquely identify a customer. Since name and address is a subset of customer data, NAML is a subset of CIML.

### NAML EXTERNAL ENTITY REFERENCE

Let me explain CIML DTD in more detail. To incorporate all the elements of NAML in the CIML DTD, we define the NAML DTD as an external entity:

```
<!ENTITY % naml SYSTEM "naml.dtd">
%naml;
```

The different possible elements of a CIML record are shown in Listing 4 and an example XML file that would be valid according to the CIML DTD is shown in Listing 5.

To date, MasterSoft has defined over 100 XML tags (excluding NAML tags) in CIML to represent customer-centric data and this is continuously evolving.

## NAML and CIML – What's Next?

MasterSoft is now actively driving the first XML standards for customer-information-quality management, namely, the NAML and CIML, to turn them into open and vendor-neutral industry standards. These

standards are now available for public review on the XML.ORG information portal ([www.xml.org](http://www.xml.org)). XML.ORG is an XML industry portal that's an independent resource for news, education and information about the application of XML in industrial and commercial settings. This XML industry portal is hosted by the Organization for the Advancement of Structured Information Standards (OASIS) and is funded by organizations that are committed to product-independent data exchange. OASIS ([www.oasis-open.org](http://www.oasis-open.org)) is the nonprofit, international consortium dedicated to accelerating the adoption of product-independent formats based on public standards. These standards include SGML, XML, HTML and CGM as well as others that are related to structured information processing. Members of OASIS are providers, users and specialists in the technologies that make these standards work in practice.

To turn NAML and CIML into industry standards, the support of an international standards body is mandatory. MasterSoft is working closely with OASIS, and a technical committee on customer information quality, which I chair, has now been formed by OASIS to achieve this.

MasterSoft is also working with the Customer Profile Exchange (CPExchange) Network, a volunteer organization dedicated to devel-

oping an open standard to facilitate the exchange of privacy-enabled customer information across enterprise applications. The CPExchange Network is hosted by the International Digital Enterprise Alliance (IDEAlliance – see [www.idealliance.org](http://www.idealliance.org)), a nonprofit, vendor-neutral organization dedicated to the development and implementation of open interoperability standards. The CPExchange Network is looking into the possibility of incorporating the NAML and CIML specifications as part of their overall specifications for customer profile exchange. ☛

## AUTHOR BIO

Ram Kumar, technology manager of Cognito, Inc., is responsible for providing both the company's technology vision and the practical application of the vision. His expertise includes middleware, e-business, intelligent systems and data quality. Ram is the author of over 60 technical papers that have been included in refereed international journals and conferences and chairs the technical committee on customer information quality recently formed by OASIS.

R.KUMAR@MSI.COM.AU

### LISTING 1

```
<?xml version="1.0"?>
<!DOCTYPE naml SYSTEM "naml.dtd">

<naml>
  <record>
    <person_details>
      <title>Sir</title>
      <first_name>William</first_name>
      <middle_name>John</middle_name>
      <middle_name>James</middle_name>
      <last_name>Smith</last_name>
      <suffix>JR</suffix>
    </person_details>
  </record>
</naml>
```

### LISTING 2

```
<?xml version="1.0"?>
<!DOCTYPE naml SYSTEM "naml.dtd">

<naml>
  <naml_record>
    <person>Sir William Smith</person>
    <address_addr_type="residence">
      2/2 Green Place
      Rockwood Park
      NSW 2060
      Australia
    </address>
    <address_addr_type="C/O">
      14 Brown Crescent
      Riverwood
      NSW 2210
      Australia
    </address>
  </naml_record>
</naml>
```

### LISTING 3

```
<address_details>
  <street_address_details>
    <sub_dwelling_details>
      <sub_dwelling_type>Unit
    </sub_dwelling_type>
    <sub_dwelling_number>2
  </sub_dwelling_details>
  <street_details>
    <street_number>2
  </street_number>
    <street_name>Green Place
  </street_name>
```

```
</street_details>
</street_address_details>
<locality>Rockwood Park</locality>
<state>NSW</state>
<postcode>2060</postcode>
<country>Australia</country>
</address_details>
```

### LISTING 4

<!--The root element CIML contains the element RECORD. All other elements are then defined within the RECORD element. This is so multiple records can be in the one XML file. -->

```
<!ELEMENT ciml record+>
<!ELEMENT record
  (unique_identifier?, naml?,
  company_trading_name*,
  company_registration_number?,
  company_tax_number?,
  (company_format_on_date|
  company_format_on_date_info)?,
  (date_of_birth|
  date_of_birth_info)?,
  age?, age_range?, gender?,
  marital_status?,
  citizenship*, country_of_birth?,
  (passport*|passport_info)?,
  religion?, ethnicity?,
  (mobile*|mobile_info)?,
  (telephone*|telephone_info)?,
  (fax*|fax_info)?,
  pager*, (email*|email_info)?,
  (url*|url_info)?,
  (credit_card*|
  credit_card_info)?,
  person_identification_number?,
  (drivers_license*|
  drivers_license_info)?,
  (frequent_flier*|
  frequent_flier_info)?,
  (person_tax_number*|
  person_tax_number_info?))>
```

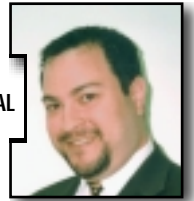
### LISTING 5

```
<?xml version="1.0"?>
<!DOCTYPE ciml SYSTEM "ciml.dtd">

<ciml>
  <record>
    <unique_id>AUS12345678</unique_id>
  </naml>
```

```
<person_details>
  <title>Mr</title>
  <first_name>Ram</first_name>
  <middle_name>Laxman
  </middle_name>
  <middle_name>Bharathan
  </middle_name>
  <last_name>Kumar</last_name>
</person_details>
<address_addr_type="residence">
  2 Green Place
  Rockwood Park
  NSW 2060
  Australia
</address>
</naml>
<date_of_birth>22 Jan 1977
</date_of_birth>
<age>23</age>
<telephone_details>
  <country_code>61</country_code>
  <area_code>2</area_code>
  <number>96756000</number>
</telephone_details>
<credit_card_info>
  <credit_card_details>
    <credit_card_type>Visa
  </credit_card_type>
    <credit_card_number>
      4509 4567 2012 3244
  </credit_card_number>
    <credit_card_expiry_date>
      April 2000
  </credit_card_expiry_date>
  </credit_card_details>
</credit_card_info>
<frequent_flier_info>
  <frequent_flier_details>
    <alliance>One World</alliance>
    <frequent_flier_number>99405678
  </frequent_flier_number>
    <date_of_joining>5 December 1995
  </date_of_joining>
  </frequent_flier_details>
</frequent_flier_information>
</record>
</ciml>
```





*Gathering and using existing legacy metadata  
with XML will increase competition*

# Conquering the E-Commerce Landscape with XML: Mastering Semantics

The term *e-commerce* (or e-everything for that matter) is now recognized by corporate executives and customers alike as denoting a new frontier. Here services like customer support, online purchasing and simple information retrieval can provide the gold standard of a global consumer society: instant gratification.

Remote services are nothing new: for years, hundreds of businesses have successfully shared and exchanged information with each other and their consumers, using a variety of media from pigeons to ponies. However, the high level of necessary human involvement have traditionally kept return on investment (ROI) on these types of services to a minimum, often with the result that companies built only limited infrastructures for such services.

The e-commerce revolution began not in the 1990s, as reports in the mainstream media would suggest, but in the 1960s, when the introduction and evolution of EDI (electronic data interchange) fundamentally altered the way large corporations did business. Originally EDI had an impact on large corporations only. Small- and medium-sized corporations were left out of the party, because the cost of implementing and maintaining EDI systems and conducting EDI transactions was high. Traditionally, EDI occurs using special ISPs – Value Added Networks (VANs) – that charge on a per-transaction basis. These charges can prove prohibitive for smaller corporations. EDI software and personnel can also prove costly for companies to acquire and retain.

The introduction of hypertext technologies to leverage the Internet has ushered in a second phase in e-commerce. Lured by the near-ubiquity of customer access and the lower costs of HTTP-based systems, large and small corporations alike have been imple-

menting business-to-customer e-commerce initiatives. But the problem that EDI was created to address – the larger challenge (and larger market) of business-to-business e-commerce – has remained unsolved by HTTP technologies. Until now. Like the hub of a wheel whose spokes have been moving inward, XML finally provides a means for unifying disparate systems and creating true information integration among businesses.

With a means of integration now available, the focus shifts to the information being shared. Here, with regard to e-commerce between businesses, the same problem exists whether you're using EDI and X12 or the Web and XML. This problem is one of semantics, not syntax. That is, businesses need to agree on common meanings for data in order to share and exchange it.

Large corporations have spent years trying to settle on standardized EDI vocabularies. The specter of this laborious effort recurring in the XML space is one of the most commonly noted reservations about XML. Fortunately, the semantic vocabularies of the numerous EDI libraries, as well as the legacy metadata of corporations, provide excellent places to start standardizing for XML. We can expect that the judicious reuse of existing vocabularies, in tandem with mapping tools, will enable XML-based solutions to skirt the time-consuming consensus-gathering process on which EDI-based systems have depended.

The next thrust in business-to-business e-commerce will be revisiting legacy applications to extract their rich metadata, then combining it with outgoing electronic communiqués to trading partners. For example, current EDI standards are highly compressed messages that contain critical business transaction data. By reexamining them and injecting them with human-readable terms – the metadata – we create a new and more powerful representation of each message. We can thus provide increased value along the processing chain by allowing humans as well as machines to:

- Detect inconsistencies in messages more easily
- Query data for statistical information that can be used to better control the flow of goods and services across the supply chain
- Perform enhanced validations to ensure business agreements are more accurately met
- Facilitate the incorporation of ever more trading partners at a much lower cost

## What Is Metadata?

Traditionally, metadata is defined as data that describes other data. For example, data within a purchase order may include a price, such as 12.94. Currently we allow our trading partner relationships to define the context against which this number will be evaluated. When dealing with our U.S. partners, we'll assume we're discussing U.S. dollars, and when dealing with our French partners, we may assume 12.94 is French francs. In this case U.S. dollars and French francs are metadata that help define how to interpret a basic decimal quantity.

# OMG

[www.omg.org](http://www.omg.org)

Due to early limitations in computing, we had to settle for allowing the trading partner context to define how to interpret a piece of data, such as price. But with advancements in technology, it's now reasonable to precisely state metadata and data within the same purchase order document.

Listing 1 shows a sample EDI X12 850 purchase order.

To the untrained eye this purchase order is extremely difficult to understand. The reason is that the fields have all been labeled with mnemonics that only partially represent the data. Yet EDI is ripe with metadata: it's trapped inside the implementations of EDI systems within companies. By translating this document with reference to existing repositories of meaning, we can re-create it in reusable form.

To illustrate, Listing 2 provides a portion of a transformed document in which the metadata has been reinstituted. This one segment of the original EDI document can now be read and processed more easily by humans and machines alike.

## AUTHOR BIO

JP Morgenthal, eVP of XMLSolutions Corporation, has been involved with the computer industry since 1986 and is a recognized leader in the industry. He was recently named vice chairperson of the Enterprise Integration Council.

## The Impact of XML

Like many new Internet technologies, XML has received generous media attention that's as likely to confuse as to excite. However, many corporate managers aren't taking the wait-and-see attitude they've adopted in the past toward other evolving technologies: there's little hesitation over whether XML is just a fad or here to stay. The reason so many people are jumping into the XML pool with both feet is that there's clear recognition that XML provides a solution to some of

the most pressing data processing problems of the twenty-first century.

As already discussed, there are two key factors for the robust exchange of information between businesses, factors that are especially critical given the general industry desire to move toward an anonymous exchange environment. The first is the need to capture and encapsulate data with its metadata so that both are readily available in the same location. The second is the need to provide a common syntax that can also express semantics – the meaning of the contained data elements. XML supplies a solution for both requirements.

The example of the transformed EDI document in Listing 2 uses XML to represent the combined set of data and metadata. Because XML is a widely accepted syntax for which many products are now starting to incorporate support, it becomes easier for people to make use of this data on a grand scale. And because XML is a hierarchical representation, it incorporates context and metadata simultaneously, which provides us with semantic value. Hence, XML offers three big wins for e-commerce:

1. XML enables companies to deliver a single package that contains data and the information necessary to process it.
2. Wide availability of XML tools, as well as XML's obvious simplicity, makes it possible for small- and mid-size businesses to participate in XML-based e-commerce.
3. The combination of the first two wins delivers a third win, which is increased competition among trading partners. This results in better customer service and pricing overall.

## Conclusions

Throughout this article we've discussed technical matters, such as how metadata assists in data processing and why XML provides an excellent representation of data to exchange with our business partners. These technical issues, however, are simply a means to an end, which is a business-to-business e-commerce environment that features low barriers to entry as well as uniform standards for data exchange. Gathering and using existing legacy metadata with XML will increase competition and ultimately provide the opportunity to deliver goods and services at a lower cost with higher quality.

There's a commonly held belief that only large corporations benefit from the automation of the supply chain, and that small- to mid-size businesses won't reap the rewards of investing in e-commerce solutions to interact with these large corporations. Such opinions are usually based on observations of the existing large-scale EDI implementations that drive most of today's business-to-business e-commerce. With the advent of XML-based business-to-business e-commerce, however, these perceptions will change rapidly. Even small- to mid-size businesses will quickly realize that by joining the pool of trading partners that support XML-based e-commerce standards, they too can purchase lower-cost, higher-quality goods and services.

And the landscape of e-commerce will change forever. ☛

JP.MORGENTHAL@XMLS.COM

### LISTING 1

```
ISA*00*          *00*          *13*RI EDI          *16*123456789
*920703*1604*U*00301*987654321*1*T>~
GS*PO*123*321*927003*1203*1112*T*004010~
ST*850*0001~
BEG*00*SA*XX-1234**19980301*AE123~
PER*BD*ED SMI TH*TE*800-123-4567~
TAX*53247765*SP*CA*****9~
FOB*PP*OR*DALLAS TX~
ITD*01*3*5**10**30*****E~
N1*ST*XMLSOLUTIONS, INC*9*123456789-0101~
N2* TRAINING DIVISION~
N3*7929 WESTPARK DRIVE, STE. 100. ~
N4*MCLEAN*VA*22182*US~
P01*1*25*EA*9.5*CT*MG*XYZ-1234~
PIDF****CELLULAR PHONE~
P01*2*75*EA*6.95*CT*MG*L505-123~
PIDF****PLIERS 8" - NEEDLE NOSE~
P01*3*48*EA*3*CT*MG*R5656-2*BP*AB123-2~
PIDF****METAL RULER - MACHINIST~
SCH*24*EA***106*19991015~
SCH*24*EA***106*19991215~
CTT*3~
AMT*TT*902.75*C~
SE*21*0001~
GE*1*1112~
```

### LISTING 2

```
<segment code="N1">
  <name>Name</name>
  <element code="98">
    <name>Entity Identifier Code</name>
    <value code="ST">Ship To</value>
  </element>
  <element code="93">
    <name>Name</name>
    <value>XMLSOLUTIONS, INC.</value>
  </element>
  <element code="66">
    <name>Identification Code Qualifier</name>
    <value code="9">DUNS with 4-character suffix</value>
  </element>
  <element code="67">
    <name>Identification Code</name>
    <value>123456789-0101</value>
  </element>
</segment>
```





# Activated Intelligence

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



*A software architecture for the design and implementation of entire distributed object systems*

## UML, MOF and XMI

In today's environment it's becoming more and more difficult to develop well-architected software for two reasons: the size and complexity of software keeps growing and the target environments are becoming more and more complex. In other words, today's environments are distributed and highly heterogeneous.

Applications are so large that one doesn't develop stand-alone systems anymore – applications are developed from existing components and are likely to use common services available in the target implementation environment. Since networks – especially the Internet – facilitate interoperability, today's applications need to interoperate with other applications and share information with them.

Through modeling, architects attempt to manage the complexity of software systems better. The model allows the software architect to separate design from implementation: it's essentially a design abstraction of the application. It doesn't capture implementation details nor does it specify interoperability semantics, information interchange formats and so on. This is clearly the limitation of the model. To provide a complete definition of the application, the architect needs to specify operational semantics, constraints and information exchange formats in addition to the design abstraction captured by the model.

The three-legged stool for software development recommended by the Object Management Group (OMG) consists of the Unified Modeling Language (UML), the Meta Object Facility (MOF) and the XML Metadata Interchange (XMI) format. The MOF is the meta-metamodel used to describe the entire software development architecture (including itself). That is, it describes modeling languages such as UML, a set of technology meta-

models such as the CORBA component model (CCM), the Enterprise JavaBeans (EJB) model and so on, as well as any other user-defined metamodels. The MOF also contains a set of rules that specify the interoperability semantics and interchange format for any of its metamodels.

To describe a software system, the software architect uses a modeling language such as UML (that's already described using MOF) to describe the "design." This design can then be enhanced using the technology models – and other models – to describe implementation/runtime semantics. The MOF rules are applied to the model to define the interoperability semantics and interchange format. Thus this architecture allows the architect to fully describe the software system.

By describing models using meta-models and describing all metamodels using a self-describing meta-metamodel, the MOF-based architecture allows applications and technologies to be described from different levels of abstraction. It also includes rules that specify interoperability semantics and interchange formats, thus facilitating interoperability and information interchange across different execution environments and technologies.

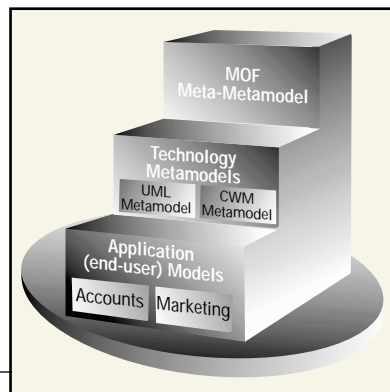


FIGURE 1 Architecture for metamodel-based ME

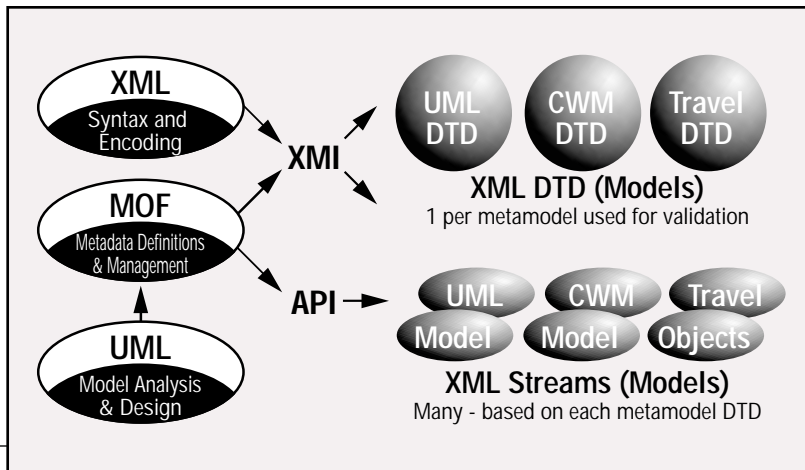


FIGURE 2 MOF model-driven software development

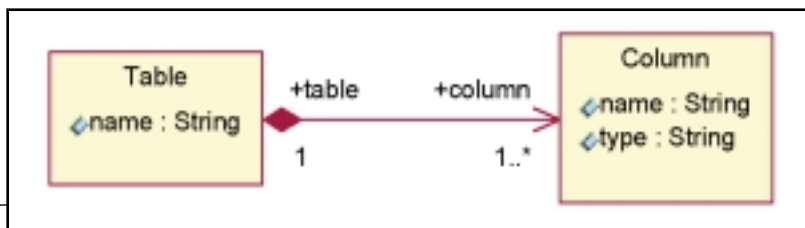


FIGURE 3 Simple RDB model

## MOF-Based Software Development Architecture

The OMG's metamodel-driven architecture consists of three layers (see Figure 1). At the top of this architecture is the MOF meta-metamodel. The MOF is the subset of UML required to describe OO systems and technologies including itself. Each modeling element in the MOF is well defined in terms of its semantics such that the MOF is capable of providing precise definitions of OO systems and technologies. Note that the UML Revision Task Force (RTF) has recommended that the "core" of UML be aligned with the MOF in a future revision of UML.

In the middle tier the MOF is used to describe the different technologies within the object management architecture (OMA). The OMA is the OMG's distributed object framework. This layer consists of a set of metamodels that describe technologies, services and business models (or information models in general) such as UML (i.e., the UML metamodel), CCM, the Common Warehouse Model (CWM), the EJB model and so on. Note that the technology models are shared by tools and applications. The technology layer is used to provide a rigorous definition of the OO modeling/implementation environment.

Models of applications and their implementation details are captured in the final tier of this architecture. The

technology models of the middle tier are used to capture different aspects of the application. Let's take, for example, an accounting application whose business logic is captured in a UML model. In this architecture the model will be an instance of the UML metamodel. Note that this model captures only the business logic of the application – it doesn't capture any implementation details. The accounting application may be targeted toward one or more implementation environments. The implementation details will be captured by the respective technology metamodels – e.g., EJB and/or CCM metamodels. In addition, the application may be developed using already available components and may make use of services available in the target environment. This information can also be captured using the respective technology metamodels. Thus a rigorous description of the application that includes its business logic as well as implementation details is captured by this three-tiered, metamodel-based approach.

For today's e-business systems (i.e., integrated systems across the enterprise) the metamodel-driven architecture described above combines the business model and the metadata captured in the "meta" levels of the business model to bring about new levels of

abstraction, personalization and extensibility of these business models – essential ingredients to manage business information (content) and application/business processes that use this content in the distributed OO environment.

## Interoperability and Information Exchange

In addition to providing a rigorous description of OO systems, the MOF contains a set of rules that define the interoperability semantics and information (metadata) exchange formats for a given information model (see Figure 2). The MOF to IDL (Interface Definition Language) transformation rules can be applied to any metamodel to produce a well-defined API. This API can be used to manipulate instance data corresponding to the respective metamodel. This API also provides introspection (a.k.a. reflection) – the ability to discover information contained in the respective metamodel. In addition to the API, the MOF rules also define the DTD corresponding to the metamodel. Any instance data of a metamodel can be represented by an XML Metadata Interchange (XMI) stream that conforms to the corresponding DTD.

Next, a simple example is used to illustrate the interoperability and information exchange capabilities of this MOF-based architecture. In this example the MOF is used to define a relation database (RDB) technology metamodel. The MOF rules are then applied to the RDB metamodel to generate the normative IDL interfaces and XMI DTD for information exchange.

## The Metamodel

This simple RDB model consists of two classes: Table and Column (see Figure 3). The Table class has one attribute named "name" of type String. Column has two attributes, "name" and "type" (which describes the data type of the column), both of type String. The model also defines an association with the following semantics:

1. Each column belongs to one and only one table.
2. Each table has one or more columns.

## THE DTD

The DTD for the simple RDB model generated using the MOF is given in Listing 1. Note that this DTD makes use of XML namespaces (e.g., xmlns:RDB). Any instance data rendered in XMI corresponding to this RDB metamodel will conform to this DTD.

## AUTHOR BIOS

*Sridhar Iyengar leads the technology strategy for object technology products at Unisys Corporation. The chief architect of the OMG MOF and XMI, Sridhar represents Unisys on OMG's board of directors. He is a frequent speaker at industry conferences.*

*Ravi Dirckze is a member of the advanced technology team at Unisys. His research interests include interoperability and integration of heterogeneous systems, metadata-based middleware, object technology and mobile computing.*

*Donald E. Baisley is senior architect of object technology at Unisys. Don works with several Unisys development projects and also with OMG on standards for XMI, UML, Data Warehousing and MOF.*

## THE INTERFACES

Due to space limitations, the entire set of interfaces generated isn't included – only the interface generated for the Table (see Listing 2). Note that the interface provides methods to get (name()) and set (set\_name()) the name of the table, as well as methods to add modify and delete columns from the table. These generated interfaces can be used to (programmatically) manipulate instance data of the RDB metamodel.

## THE XMI STREAM

Listing 3 shows the XMI stream that represents a simple table named "PERSON." This table consists of two columns:

1. "name" of type "String" used to store the name of a person
2. "socialSecurityNumber" of type "CHAR(9)" used to store a person's social security number as a nine-character string

## Current State of the Technology

As part of the ongoing process of defining the OMA, many technology models are being defined using the MOF. The CCM and the CWM are defined using it, and the MOF APIs for

interoperability and the XMI DTDs for information exchange are part of the published standard. In fact, XMI DTDs for the common warehouse process of the Metadata Coalition (MDC) have also been generated using the MOF.

At an OMG meeting in November 1998, before XMI became an OMG standard, nine products from five vendors were demonstrated working together using XMI. In fact, a demonstration on the exchange of metadata between CWM repositories of different vendors is scheduled for the September 2000 meeting of OMG to be held in Burlingame, California.

Although the MOF is an OMG standard (and generates only IDL interfaces), it's gaining acceptance outside the OMG as well. Currently, normative Java interfaces to the MOF are being defined as part of JSR-40 – the Java Community Process (JCP) Metadata API Specification. JSR-40 is scheduled to be completed in third quarter 2000.

## Summary

UML is used extensively to model object systems. It can't, however, capture implementation details, interoper-

ability semantics, information exchange formats and so on. In today's heterogeneous distributed environment, integration, interoperability and information exchange are core requirements of any software system and need to be represented in the definition of the system. The OMG MOF-based architecture blends metadata with UML and XML and allows the modeler to provide an extensive definition of a system that includes integration and interoperability APIs as well as information exchange formats. ☛

## XML References

1. Iyengar, S. (1998). "A Universal Repository Architecture using the OMG UML and MO." *Proceedings of the EDOC*, La Jolla, CA, October.
2. Kobryn, C. (1999). "UML 2001: A Standardization Odyssey." *Communications of the ACM*, Vol. 42, No. 10, October.

SRIDHAR.IYENGAR2@UNISYS.COM

RAVI.DIRCKZE@UNISYS.COM

DONALD.BAISLEY@UNISYS.COM

### LISTING 1

```
<!ENTITY % fixedDTD SYSTEM "XmiFixed.dtd">
%fixedDTD;
<!ATTLIST XMI xmlns:RDB CDATA #IMPLIED >
<!-- PACKAGE: RDB: SimpleRDB -->
<!-- ***** RDB: tableHasColumn ***** -->
<!ELEMENT RDB:Table.column ( RDB:Column)* >
<!-- CLASS: RDB:Table -->
<!ELEMENT RDB:Table.name (#PCDATA|XMI.reference)*>
<!ENTITY % RDB:TableProperties '((RDB:Table.name)?)' >
<!ENTITY % RDB:TableCompositions '(RDB:Table.column*)' >
<!ENTITY % RDB:TableAttPropsList 'name CDATA #IMPLIED' >
<!ELEMENT RDB:Table ( %RDB:TableProperties;
, (XMI.extension*)
, %RDB:TableCompositions; )?>
<!ATTLIST RDB:Table %RDB:TableAttPropsList; %XMI.element.att;
%XMI.link.att; >
<!-- CLASS: RDB:Column -->
<!ELEMENT RDB:Column.name (#PCDATA|XMI.reference)*>
<!ELEMENT RDB:Column.type (#PCDATA|XMI.reference)*>
<!ENTITY % RDB:ColumnProperties '((RDB:Column.name)?
, (RDB:Column.type)?)' >
<!ENTITY % RDB:ColumnAttPropsList 'name CDATA #IMPLIED
type CDATA #IMPLIED' >
<!ELEMENT RDB:Column ( %RDB:ColumnProperties;
, (XMI.extension*) )?>
<!ATTLIST RDB:Column %RDB:ColumnAttPropsList;
%XMI.element.att; %XMI.link.att; >
```

### LISTING 2

```
interface Table : TableClass
{
    string name ()
        raises (Reflective::MofError);
    void set_name (in string new_value)
        raises (Reflective::MofError);
```

```
ColumnSet column ()
    raises (Reflective::MofError);
void set_column (in ColumnSet new_value)
    raises (Reflective::MofError);
void add_column (in SimpleRDB::Column new_element)
    raises (Reflective::MofError);
void modify_column (
    in SimpleRDB::Column old_element,
    in SimpleRDB::Column new_element)
    raises (Reflective::NotFound, Reflective::MofError);
void remove_column (in SimpleRDB::Column old_element)
    raises (Reflective::NotFound, Reflective::MofError);
}; // end of interface Table
```

### LISTING 3

```
<?xml version="1.0" encoding="UTF-8" ?>
<XMI xmi:version="1.1">
  <XMI.header>
    <XMI.metamodel xmi:name='SimpleRDB' xmi:version='1.0' />
  </XMI.header>
  <XMI.content>
    <RDB:Table name='PERSON'>
      <RDB:Table.column>
        <RDB:Column name='name' type='String'\>
          <RDB:Column name='socialSecurityNumber'
            type='CHAR(9)'\>
            <RDB:Table.column>
          </RDB:Table>
        </XMI.content>
      </XMI>
```



# **XML DEVCON 2000**

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



JavaCo  
p/

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

n 2000

/u

on2000.com



*Eliminate the coercion from e-commerce...  
and fragmentation becomes a nonissue*

# XML: Extensibility or Fragmentation?

In my last column (*XML-J*, Vol. 1, issue 1) I talked about XML's extensibility and how it's the key to building dynamic systems. But that begs the question: Does the freedom to extend a data structure create new opportunities, or is it another example of flexibility run amok?

The debate over supporting extended or XML data structures has been taken one step further to include support of different schemas – or vocabularies. The proliferation of XML vocabularies (and sometimes competing ones) has many people worried about fragmentation. But fear not.

## What to Do About All Those Vocabularies...

If XML is supposed to be the universal common language so that all systems can talk to each other, what's the deal with all these *vocabularies*? Doesn't that defeat the purpose?

As I see it, we're at a crossroads. The development community can take the traditional route and conclude that we must all agree on vocabularies, or it can decide that managing different vocabularies could actually be a good thing for pushing the creative boundaries of what XML can do.

As history has shown us, the well-trodden path will have a predictable outcome with most of the same winners. Certain vendors will try to co-opt the standards into the ones they decide to support and corporations will still customize them anyway. Just consider how

many organizations customize EDI – and EDI is horrendously difficult to customize. Imagine what will be done with XML!

The alternative is to think somewhat orthogonally about the problem, beginning with Step One – the acceptance that vocabularies differ. Too simple? It's not a fact of life that we all have to swallow grudgingly; rather, it's an opportunity for us to use an enabling technology to do things that are truly different. Using, customizing and extending different vocabularies should be encouraged – and leveraged. In short, we should agree that we'll disagree on a single standard for XML. The trick then would be to make sure your systems are designed to handle the situation.

## Technical Reasons

Let's first start with the mundane technical reasons why XML data should be customized at will. I say they're mundane because we tend to lose sight of the key benefits of XML and assume that with enough blood, sweat and tears anything is possible. As a universal data format, XML is designed to be contorted to support different schemas because:

1. **It's easy:** Unlike EDI, a beginner with a text editor can extend an XML data structure.
2. **Apps won't break:** If your app isn't interested in a new or unrecognized element, it can simply ignore it. No lost pointers, and no memory leaks – no problem.
3. **Elements are identifiable:** Because all elements are delimited with self-identifying tags, it's easy, for example, to parse XML data to find the "price" of the "book" even if the book has an unrecognized structure.

For these reasons applications aren't strictly dependent on a predetermined schema in order to make sense of XML data, a relaxation of the rules that opens new doors for schema and vocabulary independence. And isn't that flexibility what we've all been pushing for?

## Market Forces

For all the reasons listed above, XML has found its primary residence as an enabler for today's business-to-business communication. It has been helped by the fact that EDI had already laid the foundations for the practice of handling business transactions electronically to save time and money. I think we must therefore look toward the B2B market to see how XML will be used best.

## AUTHOR BIO

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

Of course, "market forces" is just a euphemism for "Money Talks," and those with big voices like to do things their own way. You can be sure there'll be many vocabularies, and customized vocabularies on top of those. The Wal-Marts of the world will likely create their own Wal-Mart-XML (for example), no matter what standards prevail. That's not a bad thing; it's what XML is made to do. The last thing we want is for an XML standard to dictate a least-common-denominator approach to business communication.

Another market force that encourages the use of many different vocabularies is the competitive pressures generated by our Internet-driven global economy. As customers' expectations of service levels continue to rise, companies no longer have the luxury of passing on the costs of inefficiencies to their customers in the form of higher prices. And since in-depth cost analysis is only a click away for many e-customers, suppliers have to fine-tune their processes like never before.

What this all means is that Wal-Mart may not always be able to rely on their favorite suppliers – the ones that were willing to adopt Wal-Mart-XML – but may need to rely on a new supplier, a mom-and-pop shop or maybe a trading hub as their business criteria vary from week to week. For this reason planning to be locked into a single vocabulary – and therefore being locked into limited partners – is to steer your enterprise toward the B2B off-ramp.

Basically, customized vocabularies are a good thing and provide a great opportunity for those companies who realize this as they integrate their e-business enterprises. Clearly the ability to be able to work with a larger number of partners – including ones not closely held – is the wave of the future. To do this in the most effective way, and if the free market has any say about it, XML must be used in a way that is vocabulary-agnostic.

### Balancing Independence and Unity

In order to take advantage of the extensibility and new and emerging vocabularies of XML, it's critical to build systems that have schema flexibility built in. To properly "future-proof" your IT environment, your infrastructure must be designed to handle loosely coupled systems to preserve independence while unifying them to work toward a common goal.

Otherwise you have an architecture that requires co-opting any system, application or device it connects to. The last thing you want to do is place barriers-to-entry in front of partners who could possibly benefit your business. If you can eliminate the coercion from e-commerce, fragmentation becomes a nonissue. ☻

 [COCO@EXCELONCORP.COM](mailto:COCO@EXCELONCORP.COM)

## Meet JDJ EDITORS AND COLUMNISTS

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

# JavaCON 2000

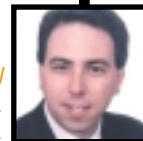


September 24-27, 2000

Santa Clara Convention Center  
Santa Clara, CA

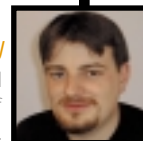
**Sean Rhody** Editor-in-Chief, *JDJ*

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



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

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



**Jason Westra** EJB Home Columnist, *JDJ*

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



**JAVA** DEVELOPERS JOURNAL

**CAMELOT** TECHNOLOGIES

## ADVERTISERS INDEX

ADVERTISER	URL	PH	PG
ACTIVATED INTELLIGENCE	WWW.HEADLINEWATCH.COM	800.455.3180	49
CAPE CLEAR	WWW.CAPECLEAR.COM	353.1.241.9900	35
CAREER OPPORTUNITIES		800.846.7591	62-65
COMPUTERWORK.COM	WWW.COMPUTERWORK.COM	800.691.8413	21
GCA	WWW.GCA.COM	703.519.8160	13,23,27
GEEK CRUISES	WWW.GEEKCRUISES.COM	650.327.3692	31
IBM	WWW.IBM.COM/DEVELOPERWORKS	800.772.2227	68
INFOSHARK	WWW.INFOHARK.COM	888.DATASHARK	9
IXIASOFT	WWW.IXIASOFT.COM	514.279.4942	19
JAVACON 2000	WWW.JAVACON2000.COM		55,54
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	58,59
OMG	WWW.OMG.ORG	781.444.0404	47
ONDISPLAY	WWW.ONDISPLAY.COM	925.355.3200	3
SD EXPO	WWW.SDEXPO.NET		37
SEQUOIA	WWW.XMLINDEX.COM	410.715.0206	17
SILVERSTREAM	WWW.SILVERSTREAM.COM	978.262.3000	67
SIMPLEX KNOWLEDGE COMPANY	WWW.SKC.COM		39
SOFTQUAD	WWW.SOFTQUAD.COM	416.544.9000	2
SOFTWARE AG	WWW.SOFTWAREAG.COM/TAMINO	925.472.4900	29
SYS-CON MEDIA	WWW.SYS-CON.COM	800.513.7111	20
WROX PRESS	WWW.WROX.COM	44.0.121.687.4100	4
XML BOOTCAMP			43
XML DEVCON 2000	WWW.XMLDEVCON.COM		53

# JDJ S

[www.jdjs](http://www.jdjs)



# Store

store.com

## XMLSolutions Delivers XML-Based Prototype for Envera Marketplace

(McLean, VA) – XMLSolutions Corporation has announced its long-term participation in the planning, architecture, development and deployment of the Envera Clearinghouse, a global electronic marketplace for B2B transactions and services. ☛

[www.envera.com](http://www.envera.com)  
[www.xmls.com](http://www.xmls.com)



## eHelp Corporation Unveils Open XML Standard for Web Usability Analysis

(San Diego, CA) – eHelp Corporation has introduced the ALURe specification, an open implementation of XML that promises



to significantly reduce abandonment



ment rates on Web sites and to boost e-commerce sales. ALURe (Aggregation and Logging of User Requests) tracks and reports usage patterns in Help and customer assistance systems on Web sites. Using this information, companies can make significant improvements to their Web sites – leading to increased completed transactions and higher revenues. ☛

[www.alurexml.org](http://www.alurexml.org)  
[www.ehelp.com](http://www.ehelp.com)

## Software AG USA Signs On Treehouse Software as XML Distributor

(Walnut Creek, CA) – Software AG, Inc. USA and Treehouse Software, Inc., have entered into a

distribution partnership allowing TSI to market, sell and support Software AG's native



XML e-business products in the United States. TSI will immediately begin selling Software AG's hallmark XML offering,



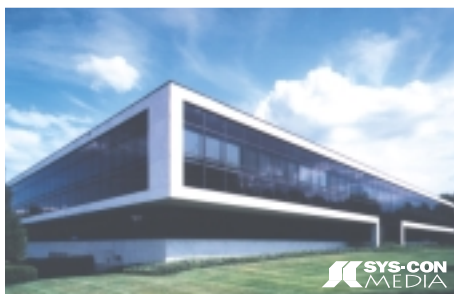
Tamino – the world's first native XML information server. ☛

[www.softwareagusa.com](http://www.softwareagusa.com)  
[www.treehouse.com](http://www.treehouse.com)

## SoftQuad Releases MarketAgility

(Toronto, ON) – SoftQuad Software, Ltd., has announced MarketAgility, an XML-based content solution that gives businesses more power and control over the creation, management and real-time delivery of product information to e-marketplaces and e-procurement systems. MarketAgility provides suppliers with an efficient and cost-effective way to move product information from their enterprises to multiple electronic distribution channels. ☛

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



## SYS-CON Media Expands

(Montvale, NJ) – SYS-CON Media, Inc., publisher of *XML Journal* and other e-business/Internet and Web-related magazines such as *Java Developer's Journal*, has relocated its worldwide corporate headquarters from Pearl River, New York, to nearby Montvale, New Jersey.

## JDOM Provides Fast, Easy Access to XML from Java

(Sebastopol, CA) – JDOM (Java Document Object Model) is a new technology that enables Java developers to read, change and write XML data more easily than before. Created by Jason Hunter and Brett McLaughlin, JDOM has just been released under an open

source license. A Web site dedicated to promoting

the understanding and use of JDOM is available at

<http://jdom.org> ☛

## XMLSolutions Launches Support for xCBL

(McLean, VA) – XMLSolutions Corporation is introducing XEDI for xCBL, an out-of-the-box solution that enables B2B e-commerce



across the supply chain utilizing xCBL from Commerce One, Inc.

The integration of XEDI with xCBL provides every company currently utilizing EDI with the ability to connect to xCBL-enabled applications. ☛

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

SYS-CON now occupies space in the 66,000 square-foot building pictured at left, situated in prestigious northern Bergen County. New York City is just a short drive away via the George Washington or Tappan Zee bridges, and the new location is convenient to Newark International Airport via a local shuttle service.

Prominent corporate neighbors within a half-mile distance include Sony, Mercedes and Volvo (U.S. headquarters), A&P and Grand Union (world headquarters), and other national and international corporations. ☛

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

## SoftQuad Partners with Extensibility, Inc.

(Toronto, ON) – SoftQuad Software, Ltd., and Extensibility, Inc., have announced that SoftQuad's XMetaL, a content creation solution, will be



integrated with a fully functional trial version of Extensibility's XML Authority, a leading schema design and management solution. ☛

[www.extensibility.com](http://www.extensibility.com)  
[www.softquad.com](http://www.softquad.com)

## Major Companies Join Hands to Force Net Standards

Some of the biggest players in the industry are forming a consortium to set standards for Web applications that would supersede Java and other emerging Internet standards – including HTML, XML and HTTP.

IBM, Intel, Oracle, Hewlett-Packard and Compaq have

joined with several smaller companies to form

OpenServer.org, which plans to create

"a vendor-neutral environment" to

ensure the compatibility of applications developed

according to open Internet

standards. ☛

[www.OpenServer.org](http://www.OpenServer.org)

## Delano Launches Component Pack for XML

(Toronto, ON) – Delano Technology Corporation has announced the release of the Delano Component Pack for XML, which extends the

Delano e-Business Interaction Suite to provide

native support for the XML and the standard XML document formats used in B2B applications. ☛

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



## PurchasingCenter.com Selects Mercator

(Wilton, CT) – PurchasingCenter.com, Inc., a Web portal to the industrial marketplace, has chosen Mercator Commerce Broker as the e-business integration platform for its maintenance, repair and operating e-procurement marketplace. Mercator Commerce Broker will provide




PurchasingCenter.com with a scalable integration architecture to manage its supply chain, streamline order management and reduce costs associated with the procurement process.   
[www.purchasingcenter.com](http://www.purchasingcenter.com)  
[www.mercator.com](http://www.mercator.com)

## eXcelon Corp Introduces Two New E-Products

(Burlington, MA) – eXcelon Corporation, a leading provider of dynamic B2B solutions, has introduced two new products that lower the barrier to participating in B2B e-commerce and creating information-rich B2B portals.



eXcelon's B2B Portal Server is a new platform that automatically collects and publishes business content from anywhere in a partner network to a central, interactive e-business site.

Xpress, part of the eXcelon B2B Integration Server product family, is a new B2B technology that quickly connects an organization to thousands of partners, creating cross-organization business processes that increase revenue, improve operational efficiency and allow end-to-end visibility and control of business operations. 

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

## Oracle Announces iFS

(Redwood City, CA) – Oracle Corp. has introduced its latest product designed to encourage people to store information on the Internet instead of on their personal computers. Dubbed the Internet File System, or iFS, the software allows personal computer hard-drive files to be transferred to an Internet server. Once the information is put in this electronic storehouse – which Oracle calls the “o-drive” – it


becomes part of a database that can be quickly searched.

Oracle's CEO, Larry Ellison, has described the iFS as one of the most significant challenges to Microsoft's dominance since Netscape developed its Internet browser in the mid-1990s. The company plans to provide the product free to customers who have already purchased Oracle8i database software.

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

## eXcelon Corporation Launches Industry's First Complete B2B E-Commerce Solution


(Burlington, MA) – eXcelon Corporation has introduced the industry's first complete set of products and services empowering organizations to join business processes with their partners over the Internet.

The complete line of eXcelon B2B technology includes the B2B Integration Server, the B2B Portal Server, the B2B Partner Server and Xpress. 

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

## HiT Software Launches Allora

(San Jose, CA) – HiT Software introduces Allora, Web interoperability middleware that gives application developers high-performance XML access to relational databases. HiT Allora gives IT organizations the


advantage of a consistent XML data access model, synchronous SQL transaction access, asynchronous message queue access and significantly higher performance compared to generic DOM development. Allora is a World Wide Web Consortium ([www.w3c.org](http://www.w3c.org)) DOM implementation designed specifically for relational database access. 

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

## Arbortext to Provide XML-Based Content Management Software Using Oracle iFS

(Ann Arbor, MI) – Arbortext, Inc., a leading provider of XML-based e-content software, has announced the availability of Arbortext's Epic Editor and Epic Editor LE – now with new support for the Oracle Internet File System (iFS).



Oracle iFS is the first file system built for the Internet and combines the simplicity of the file system with the powerful information management capabilities of Oracle8i. 

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


## Icon Releases XML Spy 3.0

(Vienna, Austria) – Icon Information Systems has released XML Spy 3.0, the first true IDE for XML that includes XML editing and validation, Schema/DTD



editing and validation, and XSL editing and transformation.

XML Spy 3.0 fully supports DTDs, Document Content Description (DCD), XML-Data Reduced (XDR) and BizTalk, and already contains support for most of the new April

7 W3C XML Schemas. A free 30-day evaluation version can be downloaded from the XML Spy Web site. 

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

## Vignette to Buy OnDisplay

(Austin, TX) – Vignette Corp., a provider of software and services to bring business operations online, is to acquire rival OnDisplay, Inc., in a stock swap valued at \$1.7 billion.



The acquisition will mean that Vignette has about 2,000 employees, 870 customers and global operations in the United States, South America, Europe, Asia and Australia. 

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

## WebMethods to Acquire Active in Stock Deal

(Fairfax, VA) – E-commerce software maker WebMethods has agreed to acquire Active Software in a stock deal valued at \$1.3 billion.

With the deal, WebMethods is aiming to offer a more complete package to help companies build online marketplaces and will therefore combine Active Software's enterprise application integration tools with its own business e-com-



merce software.

The acquisition will help expand WebMethods' growing international presence by adding offices in the United Kingdom, France, Germany and the Netherlands. The newly combined company will have nearly 600 employees worldwide. 



# Career Opp

# opportunities



# Career Opp

# opportunities

# XML: Staying in Front of Rapid Change



WRITTEN BY ALAN GOLD

We now live in an age of rapid change. Long gone are the days of writing a business plan, analyzing every detail, then living with that plan for years. The business climate of today demands split-second decisions, rapid execution and competitive leapfrogging that happens in a matter of weeks, not years. In other words, the MO for most businesses is constant change: add a feature, add a partner, add a business unit – most often outside the scope of the original plan.

The Internet is the ultimate platform for constant change and growth. It is a boundless, extensible, sprawling network that not only supports random new additions, but thrives from it. For example, creating a new Web site is completely nondisruptive to the rest of the Network, and gives all existing sites the option of linking to and leveraging the new resource. The structure of the Internet gives companies access to the universal set of options rather than to a defined and restricted subset.

XML, the language of business-to-business e-commerce, embodies the same spirit. Like the Internet itself, XML is also extensible, making it possible to add new data elements and information as changes to systems are needed. Extending an XML data document doesn't disrupt any existing system that reads the document, but gives all systems universal access to the new information – even though it wasn't part of the original message specification.

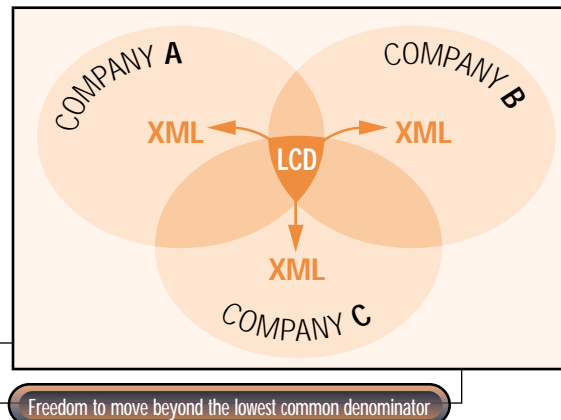
Consider a simple example. The sneaker industry probably has an XML dialect that includes data elements for the size, price, model, color and a few other predictable things. This enables shoe retailers, shoe manufacturers and shoelace suppliers to do business electronically. But what if one company wants to make an addition to their offering? What does that do to this perfect little setup?

I recently read that Nike is developing a new gimmick: they'll allow customers to customize the text written on the heel of their sneakers. Everything from pet preferences to political opinions to alma mater endorsements can now be proudly displayed on your feet, but only if you choose Nike.

A cute idea, but what does this do to the B2B network that's shared by Nike's (and perhaps Reebok's) entire supply chain? Nothing, unless your organization needs to know about this new bit of information in order to

execute. Nike can freely extend the message format shared across its trading network with a <heel-text> element. The manufacturer picks it up, but the shoelace supplier ignores it. In fact, they never even have to be told. Voilà – an easy addition brought to you by the wonders of XML.

This example demonstrates something that's more than just a convenient way of doing standards-based communication, something that's fundamentally different. We're used to industry standards that represent the lowest common denominator. Leading (and dominating) industry vendors get together, decide what they need that's common to all, then create the standard that defines the limitation that all companies must adhere to. This is equivalent to being locked into the small intersecting area in the center of a Venn diagram.



XML is unlike traditional standards in that it doesn't limit you to the lowest common denominator, but rather gives organizations access to the universal set. The industry-driven dialect is not a limit but a starting point: take the specification and extend it to include support for any unpredictable feature that your latest business decision requires.

The freedom XML gives you to add to an existing standard or message format is the key to growth. Just like the Internet, XML is a flexible and extensible technology that not only makes it possible to take an incremental approach to business strategy but encourages it. Being able to

extend a system, network or organization in unpredictable ways delivers business agility that is necessary in today's competitive environment.

The Internet is fundamentally different from a T1 line just as XML is fundamentally different from traditional standards. They support continuous growth and change, making it possible to nimbly implement strategic objectives without the threat of disruption. This enables you to keep your company ahead of the business curve. ☛

## AUTHOR BIO

Alan Gold joined eXcelon Corporation in April as chief marketing officer and vp-marketing. In his previous position at MarketMAX Alan developed the company's strategic direction and partnering strategy, and identified and cultivated key corporate alliances. The author of more than 150 articles in various publications, he has been an active participant in industry trade organizations and a frequent speaker at industry and trade conferences.

AGOLD@EXCELONCORP.COM

# Silverstream

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

# IBM

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