Content Management • Enterprise Solutions • Labs

XML JOURNAL

THE ULTIMATE XML ENTERPRISE RESOURCE

March 2004 Volume: 5 Issue: 3

XML-JOURNAL.COM

Realizes Its Potential

Adaptive XML application is ideal for managing enterprise product information 6

FREE* CD

SEE PG. 19

www.sys-con.com/FREECD

DISPLAY UNTIL May 31, 2004

$6.99US $7.99CAN

04>

0 09281 01314 3

BEA WebLogic Workshop 8.1
makes you more

AMAZING.

Development Tools
December 30, 2003
BEA WebLogic Workshop 8.1
BEA Systems, Inc.

"A shared, consistent development platform…for J2EE experts and business-oriented developers alike."

Java Pro Readers Choice Award

"WebLogic Workshop empowers all application developers…not just J2EE developers."

Software Development Jolt Product Excellence Award

"The integration of an IDE, controls and a deployment environment…greatly enhances developer productivity."

CRN Test Center Product of the Year

"By far the most innovative development tool reviewed this year…when compared with other tools, Workshop blew away the competition."

But don't take their word for it, find out for yourself now, for free.

BEA's WebLogic Workshop 8.1 has won every major award for software development tools in the past year. It's a full-featured Java development environment that lets you visually build and assemble enterprise-scale Web Services, Web Applications, JSPs, Portals, EJBs, and Business Process models based on the latest standards and open source technologies.

Get it now. For free. Visit http://dev2dev.bea.com/workshop3

dev2dev™

bea®

EDITORIAL OFFICES
SYS-CON MEDIA
135 CHESTNUT RIDGE ROAD, MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 **FAX:** 201 782-9637
XML-JOURNAL (ISSN# 1534-9780)
is published monthly (12 times a year)
by SYS-CON Publications, Inc.
Periodicals postage pending
Montvale, NJ 07645 and additional mailing offices.
POSTMASTER: Send address changes to:
XML-JOURNAL, SYS-CON Publications, Inc.,
135 Chestnut Ridge Road, Montvale, NJ 07645.
Worldwide Newsstand Distribution
Curtis Circulation Company, New Milford, NJ
Newsstand Distribution Consultant:
Gregory Associates, WRDS
732 607-9941, BJGAssociates@cs.com

FOR LIST RENTAL INFORMATION:
Kevin Collopy: 845 731-2684,
kevin.collopy@edithroman.com
Frank Cipolla: 845 731-3832,
frank.cipolla@epostdirect.com

Secure Web Services Using Identity Management

WRITTEN BY **ASHISH LARIVEE**

Web services provide the architecture for allowing different systems to interoperate. By removing many of the challenges associated with systems integration, Web services allow organizations to achieve significant business results with current systems.

To achieve these results, Web services applications must be able to communicate with each other. However, to be a viable option for widespread use in mission-critical business solutions, communication between Web services must be secure to protect the underlying business systems.

As interactions between Web service requestors and providers increase, the need for trusted relationships also increases. Consider how traditional business transactions are conducted. Two parties meet, propose a transaction, and negotiate a relationship of trust. They often develop this relationship by demonstrating the value they bring to the transaction and validating their ability to complete the transaction.

This same process applies to service requestors and providers. The requestor and provider must establish a relationship of trust, which can be done by creating an identity for users and services. After each party confirms the other's identity, they can conduct their transactions with confidence.

Using identity information to confirm the validity of a user or service is known as role-based authentication. Role-based authentication relies on the fact that a "user" has a known, defined identity in a system. The user's access to systems, resources, and information is defined by their identity. For example, a manager would have access to a different set of information and systems than an individual contributor because their roles differ. Identity management and role-based authentication are important concepts for providing the authentication, authorization and single sign-on pieces of the secure Web services puzzle.

Current implementations of this concept build mechanisms for storing identity and role information in individual Web services. This approach works relatively well for single Web services or proprietary, internal implementations. However, it does not translate to solutions where multiple Web services in internal and public systems must work in harmony to accomplish complex business processes. There is no guarantee that these discrete identity management implementations will work with each other if the Web services must interact.

The Liberty Alliance Project was started to avoid this problem. One of the most striking propositions from the Liberty Alliance is the notion of delegating authentication, access control, and identity management to specialized identity provider services. By offloading identity management tasks to a separate provider, you create a framework that simplifies developing secure Web services solutions.

In the Liberty Alliance's view of Web services architectures, identity providers offer three important, identity-based security services: authentication, authorization, and single sign-on. Identity providers handle these security services by using directories as a central repository of identity information that is used to authenticate users or services and authorize their access to services, resources, or information based on roles and privileges stored in their identity profile.

To extend this to Web services each Web service is given an identity within the directory. For complex business processes, the identity of all actors in the process must be verified – including the Web services. Because directory services can manage information about any kind of object, human or otherwise, you can maintain identity information about a Web service in a directory.

By giving a Web service an identity in a directory you can validate a Web service's identity in the same way that you would a human user's, apply the directory's built-in access controls to the Web service, and federate the Web service among many service providers for single sign-on capabilities. You also create an architecture that allows multiple Web services to interact and support complex business processes.

Directories are excellent for providing identity services as they are designed to manage internal and Web-based relationships between user identities, resources, and security policies. Perhaps most important, directory technology is here now and it's reliable. You don't have to implement new and unproven identity management technology within each Web service. Instead, you can use proven technology to provide authentication, authorization and single sign-on capabilities for your secure Web services solutions.

Author Bio

Ashish Larivee has designed and developed many enterprise applications across a variety of platforms including Microsoft, Lotus Notes/Domino, and the J2EE platform. In her current role, she helps define the strategy and product direction across Novell's Web Application Development Products.

ALARIVEE@NOVELL.COM

Assande EAI Messaging Suite features

- XML Schema Repository
- Data Abstraction, Transformation, and Mapping
- Message Assembly and Generation from Components
- Messaging Integration Service Deployment
- Automated Naming Standards, Version Control, and DIFF Functions
- Infrastructure Utilization and Pipeline Reporting
- Conversation Deployment and Message Release Management
- State Management and Integrated Workflow
- Advanced Search Capabilities
- Integration Documentation (MS Word, Excel) Repository

ASSANDÉ™
INTEGRATION MANAGEMENT COMPANY

http://www.assande.com

SYS-CON MEDIA
135 Chestnut Ridge Rd., Montvale, NJ 07645
TELEPHONE: 201 802-3000 **FAX:** 201 782-9637

PRESIDENT and CEO
Fuat Kircaali fuat@sys-con.com

VP, Business Development
Grisha Davida grisha@sys-con.com
Group Publisher
Jeremy Geelan jeremy@sys-con.com
Technical Director
Alan Williamson alan@sys-con.com

ADVERTISING
Senior VP, Sales & Marketing
Carmen Gonzalez carmen@sys-con.com
VP, Sales & Marketing
Miles Silverman miles@sys-con.com
Advertising Director
Robyn Forma robyn@sys-con.com
Director of Sales & Marketing
Megan Mussa megan@sys-con.com
Advertising Sales Managers
Alisa Catalano alisa@sys-con.com
Carrie Gebert carrieg@sys-con.com
Associate Sales Managers
Kristin Kuhnle kristin@sys-con.com
Beth Jones beth@sys-con.com

SYS-CON EVENTS
President
Grisha Davida grisha@sys-con.com
Conference Manager
Lin Goetz lin@sys-con.com
National Sales Manager
Sean Raman raman@sys-con.com

CUSTOMER RELATIONS
Circulation Service Coordinators
Shelia Dickerson shelia@sys-con.com
Edna Earle Russell edna@sys-con.com
Linda Lipton linda@sys-con.com
Merline Noel merline@sys-con.com

JDJ STORE
Manager
Brunilda Staropoli bruni@sys-con.com

WEB SERVICES
VP, Information Systems
Robert Diamond robert@sys-con.com
Web Designers
Stephen Kilmurray stephen@sys-con.com
Christopher Croce chris@sys-con.com
Online Editor
Lin Goetz lin@sys-con.com

ACCOUNTING
Accounts Receivable
Charlotte Lopez charlotte@sys-con.com
Financial Analyst
Joan LaRose joan@sys-con.com
Accounts Payable
Betty White betty@sys-con.com

SUBSCRIPTIONS
SUBSCRIBE@SYS-CON.COM
1 888 303-5282
For subscriptions and requests for bulk orders, please send your letters to Subscription Department
Cover Price: $6.99/issue
Domestic: $69.99/yr (12 issues)
Canada/Mexico: $89.99/yr
all other countries $99.99/yr
(U.S. Banks or Money Orders)
Back issues: $12 U.S. $15 all others

Cleaning Up XML

WRITTEN BY **ADAM KOLAWA**

Garbage in, garbage out – it's an axiom that applies to many aspects of enterprise development, but none more so than building reliable and robust Web applications and integration projects with XML. Since its inception, XML has been seen as the cure-all for every problem related to Web application development. However, poorly written XML can either slow down an integration project, or worse, cause the integration project to collapse.

It's important to understand some of the inefficiencies of XML, as well as how you can "clean up" and prevent the use of poorly written XML in development projects. After all, system performance is only as good as the data received and the instructions given. If errors are contained in the XML, it is more likely than not that the system will crash.

One of the main benefits of XML is that it provides mechanisms for verifying document validity through the use of Document Type Definitions (DTDs) and XML Schemas. When creating an XML document, developers can reference either of these mechanisms from within the document itself. The DTD or schema that is referenced will specify exactly how the XML document is to be processed, which elements and attributes are contained in the document, and the order in which these elements and attributes should be listed.

While referencing DTDs or schemas can guarantee the validity of XML documents, there is no requirement that developers use headers to reference DTDs or schemas at all. In fact, developers need only to follow simple syntax rules in order for an XML document to be "well-formed." However, a well-formed document is not necessarily a valid document. Without referencing either a DTD or a schema, there is no way to verify whether or not the XML document is valid. Therefore, measures must be taken to ensure that XML documents do, in fact, reference a DTD or schema.

To guarantee that an XML document references a DTD or schema, development teams can adopt a rule-based system that can detect and prevent errors within the XML code. Developers can create rules that impose constraints on XML documents to verify validity.

XML is created in plain text and utilizes actual words or phrases that have specific meanings to developers. However, even though XML can be read and written by humans, it does not necessarily mean that humans can understand XML – developers can still create unreadable XML code. An element that has a specific meaning to one developer may be of no use, or make no sense, to another developer.

To prevent ambiguous XML code, development teams must mutually agree upon a standard XML vocabulary. With a standard language in place, developers within a team will be more apt to understand each other's code. Naming conventions can be established that verify whether code follows rules that verify anything from W3C guidelines for a specific language, to team-naming standards, to project-specific design requirements, to the proper usage of custom XML tags.

Although the W3C has made an effort to establish a common language, vocabulary, and protocol for XML, these standards are still in development and are constantly changing. Companies that adhere to proposed standards that are not yet fully mature must be prepared to keep up with any changes of those standards in the future. Without any stability in XML standards, developers are forced to either keep up with the rapid changes or fall behind.

In spite of the chaos of standards that may exist for XML development, the recent release of Basic Profile 1.0 provides some guidance to developers seeking a widely used XML standard. The Web Services Interoperability Organization (WS-I) Basic Profile 1.0 consists of specifications that establish a baseline for interoperable Web services. These specifications provide a common framework for implementing XML and building integration projects. Therefore, developers can be confident that the XML standards they use will not be privy to constant flux and change if they create rules that enforce these standards.

If it is written and used properly, XML offers many advantages that can support reliable and robust Web applications. The key to successfully using XML in an integration project is to first understand the inefficiencies that may cause poorly written XML, then utilize the proper techniques that verify correctness at each level of the implementation.

Author Bio

Adam Kolawa is CEO, chairman, and a cofounder of Parasoft, a company that creates value-added products that significantly improve the software development process. Adam is a well-known writer and speaker on industry issues and in 2001 was awarded the Los Angeles Ernst & Young Entrepreneur of the Year Award in the software category.

AK@PARASOFT.COM

PIM Realizes its Potential

WRITTEN BY
SANDEEP NAWATHE

Adaptive XML application is ideal for managing enterprise product information

Product information is one of the few remaining strategic information assets within an enterprise that does not have a dedicated system of record. This is not surprising when one considers the complexities of product information management (PIM). Specifically, a comprehensive product information management application must be able to handle structured and unstructured information, share data, accommodate a variety of constantly changing data models, manage multiple inter-relationships, and enable multiple hierarchies, all without duplication of data. While enterprises and vendors alike have turned to content management and relational database architectures to address the challenges of product information management, these approaches have proven to be very limited and far less than satisfactory.

Recognizing the shortcomings of traditional approaches, several leading companies are now utilizing an emerging type of adaptive XML application to more effectively manage their product information. This new adaptive XML approach allows management of both unstructured and structured data and also provides extensive and rich functionality. Rather than limiting XML's use to inbound/outbound transformations, this approach uses XML as a core data model for the application. Traditionally, methodology for building applications utilizes compile-time data models; this makes the application rigid and tightly bound to the data model. The adaptive application utilizes an XML-based runtime data model and is used in all tiers of the application. In short, an application that is built using this approach easily handles the complexities of product data management and adapts to changing data models, which allows further extension of the capabilities beyond those of traditional applications.

Product Information Management and Its Challenges

Generally speaking, product information management (PIM) refers to a system for managing all types of information related to products. A PIM system needs to provide a system of record for products throughout the enterprise and maintain that information in a centralized location. In addition, it must provide rich functionality to address the challenges specific to product information.

Product information is unique in the variety of different data types associated with it. The information could include part number, price, and size (structured data); images, drawings, and merchandizing text (unstructured data); or inventory data (transactional data). To access, manipulate, and store each of these data types and maintain data integrity is a challenging task.

Besides managing different data types, product information systems must conform to unique and changing schemas.

To be an effective long-term solution, a PIM system must accommodate unique data models for the enterprise as well as the product and allow for changes to either of those schemas as required. Frequent changes to product information in today's rapidly evolving marketplace make application maintenance a burden on most IT resources.

Product data must be organized and classified into taxonomies that consist of an arbitrary hierarchy of categories and subcategories. With product data, it is common to have multiple simultaneous hierarchies. A PIM system needs to support these without duplication of data.

Also, products will often share data such as warranty information. This becomes a daunting manual process, particularly when shared information is changed or multiple new products are created. It is critical for a PIM system to automatically share data changes in all places where reusable content appears, thereby avoiding costly duplication, data inconsistencies, and time-consuming maintenance cycles.

The organization and sharing of information between different products and product types creates unique relationships that must be maintained within a PIM system. As product needs change, the creation of kits, bundles, and accessories is common with new, even unknown, merchandising relationships consistently being created. Referential integrity of the complex data relations must be maintained even as management operations dictate change.

Other required functionality, such as search capabilities, data validation, and synchronization with trading partners and external data stores, is indicative of the additional requirements of an effective PIM solution.

Traditional PIM Solutions

Sometimes product information management is also called product content management. This usually suggests a predisposition toward enterprise content management (ECM). An ECM system was originally designed to manage large volumes of documents and Web pages. In its use as a PIM system, product data is maintained as a set of documents, which is excellent for unstructured data. In order to manage structured data or attributes, however, the system uses name-value pairs, which is common to all content management systems. While an ECM system enables the support of a single hierarchical view of product category data in the form of folders and documents, it does not handle multiple hierarchies well; unfortunately, the support for data sharing within an ECM system requires data duplication.

A more traditional PIM system is a relational database management system (RDBMS) in which product data is stored as a set of tables and records in a single central data repository. Structured components of product data are managed well with this type of system, but unstructured items must be stored in database fields as BLOBs, limiting some field sizes. Further, an RDBMS requires predefined database schemas in the form of tables, which impose limitations on the type of product schema that can be supported. Therefore, large sets of unused fields are included in the database to account for custom data requirements and changing schemas. Hierarchy organization and complex data relations are not handled in this type of system.

Adaptive XML Application as a PIM Solution

The extensive use of XML throughout a PIM system has only recently come to the forefront as a viable and competitive alternative to traditional PIM solutions. In the past, XML was simply used for inbound and outbound transformations. Utilized in the correct fashion, however, XML can be used as the foundation for a far more powerful adaptive product information management application. This new adaptive XML approach is quickly gaining ground as the most effective means of managing product information at some of the largest and most progressive companies in the world.

Until now, structured data has commonly been represented as records in RDBMS, and unstructured data has been stored as documents. However, XML provides an ideal means of representing both structured and unstructured data, as shown in Figure 1. XML is therefore quintessential in handling complex product information that consists of both structured and unstructured information.

While the management of different data types is an important advantage in a product information management application, it is now possible to go even further with XML to create a fundamentally new approach to application architecture – an adaptive application. In an adaptive application, the three tiers adjust to runtime changes of data schemas as shown in Figure 2. Such a system has no prebuilt notion of data schemas. At runtime the appropriate schema is located and interpreted for a given piece of product information. All the computations needed for a given schema are also applied at runtime.

All tiers of an adaptive application rely on the open standard W3C DOM as the only data model. The W3C DOM is a runtime data model that represents XML DOM as the common data model across all tiers of the application. This eliminates the need for data mapping and transformation commonly found in *n*-tier applications. Also, the runtime interpretation of schema means the system can work easily with custom data models, causing a significant reduction in product implementation time.

Technical Considerations and Concepts

One of the fundamental concepts that must be changed is the notion of XML as a document versus a fragment. This is an important distinction that has a profound impact on the flexibility and functionality of an adaptive XML application. Today, XML is commonly thought of as a document with tags. This impression is problematic when it comes to storing structured data in the form of records. If a single document represents a single record, too many documents would be created and become difficult to manage. Alternatively, if multiple records were represented in a single document, then an entire set of

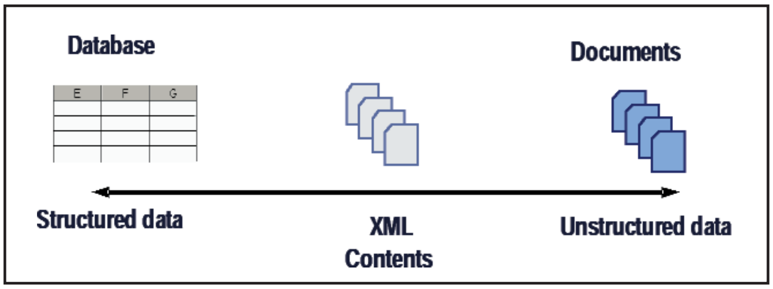


Figure 1 • XML handles structured and unstructured data

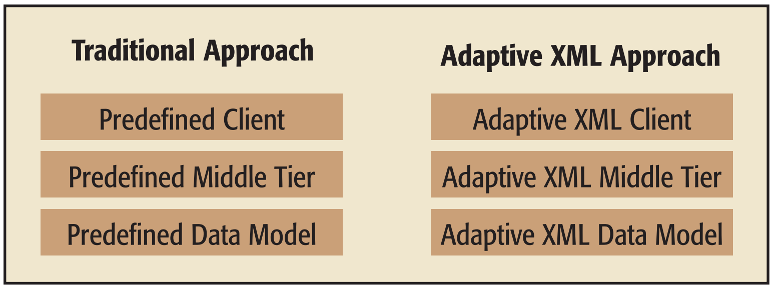


Figure 2 • Traditional application vs adaptive XML application

records would have to be manipulated each time a single document is sourced.

However, an alternative to this is the concept of XML as a fragment, with individual fragments accessed by value or reference. This enables read/write access to the fragments, with the fragments stored as either records in a database or as an XML document. This successfully manages both structured and unstructured product data whereas more traditional PIM systems falter.

Once XML is handled as a fragment, the management of these fragments must be considered. To effectively manipulate the XML fragments, W3C standard Xpointers can be used as XML fragment locators. Fragments can then be located within any system – file, content management, database, or legacy systems. The combination of XML fragments and Xpointers provides a complete uniform access layer to all enterprise data and business objects.

XML validation is another challenge that must be addressed with respect to XML fragments. In the XML world, XML schemas are used to validate XML documents. XML document validation relies on its ability to locate schemas. The schema location directives are specified at a document level. The use of XML fragments versus documents therefore creates a conundrum because there are no documents to validate. An alternative approach is to store XML schemas in a library and compute them as needed for a given XML fragment. To locate a schema for an XML fragment independent of its document, a schema locator service needs to be built. Locating the schemas for an XML fragment at runtime easily avoids hardwired data models in the application. This approach also enables schema changes to be detected at runtime, making applications resilient to data model changes. Given the constant, even unknown, changes to product information requirements, both internally and with trading partners, this ability to adapt to changing schemas is absolutely essential to providing a solution that meets the enterprise's current and future needs.

Because product information management inherently consists of taxonomies and complex data relations, XML must be able to manage these relationships effectively to be part of a successful PIM system. Xpointers can be thought of as providing the static relationship between two objects – a single pointer referencing another object. This technique can be used to manage multiple hierarchies without duplicating data. Referential integrity constraints can be easily enforced using simple link tables. Use of Xpointers in conjunction with Xlink provides considerable flexibility in managing object relationships.

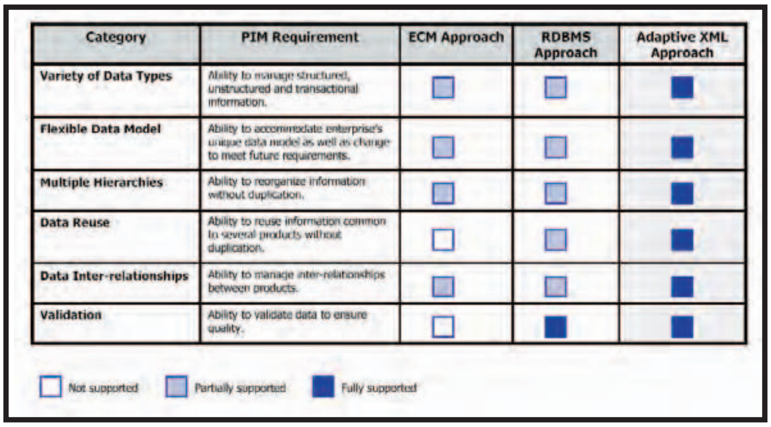
The combination also provides a powerful tool for creating and managing multiple hierarchical views as well as complex data relationships involved in merchandising and data sharing. In addition, as these relationships are not expressed in relational tables, manipulating them becomes effortless. The links and pointers can be easily created, redirected, or removed. Sharing information is similarly straightforward, as links or references are directed to existing data, preventing time-consuming and expensive data duplication within the PIM system. A change in the shared data allows all information to be updated automatically.

"An adaptive XML-based application is proving to be the future of any enterprise's successful PIM system"

An adaptive XML approach also has significant advantages in terms of the client layer. Any XML-capable authoring tool such as Excel, Word, PDF, and HTML can be used to easily interact with the application. Working with the product information management application via a spreadsheet is as simple as working through an XML-based client.

Adaptive XML vs Alternative Approaches

An adaptive application enables better management of structured and unstructured product information, as well as providing superior data model and data relationship capabilities. Table 1 summarizes how an adaptive XML approach provides superior product information management over alternative approaches across a variety of business requirements.



Category	PIM Requirement	ECM Approach	RDBMS Approach	Adaptive XML Approach
Variety of Data Types	Ability to manage structured, unstructured and transactional information.	▨	▨	■
Flexible Data Model	Ability to accommodate enterprise's unique data model as well as change to meet future requirements.	▨	▨	■
Multiple Hierarchies	Ability to reorganize information without duplication.	▨	▨	■
Data Reuse	Ability to reuse information common to several products without duplication.	□	▨	■
Data Inter-relationships	Ability to manage inter-relationships between products.	▨	▨	■
Validation	Ability to validate data to ensure quality.	□	■	■

□ Not supported ▨ Partially supported ■ Fully supported

Table 1 • A comparison summary of the ECM, RDBMS, and adaptive XML approaches to managing product information

Bottom Line

Product information requires a very specific, flexible system to enable maximum functionality in an enterprise. An adaptive XML-based application is proving to be the future of any enterprise's successful PIM system. An adaptive XML-based product information management application is able to work with both structured and unstructured information, as well as quickly and easily incorporate changing product data models, product types and complex product data relationships. This flexibility will become a required feature of any PIM system as product information requirements continue to evolve. However, it is important to be wary of architectures that claim to utilize XML in their PIM system – often it is used only on the inbound or outbound transformations, limiting the full potential that XML has to offer in meeting requirements of an effective PIM system.

Resource

- "Synchronizing Product Information Across and Beyond the Enterprise with an Adaptive XML Product Management Solution": www.fulldegree.com

AUTHOR BIO

Sandeep Nawathe is the founder and chief technology officer for Full Degree, Inc. He is a technology innovator responsible for the core technology, system architecture, and strategic technology initiatives for the world's largest transaction processing system. Sandeep recently spoke at Content World 2003 on the value of XML and schema independence.

SNAWATHE@FULLDEGREE.COM

Web Services Help

Now

LEARN

DEBUG

TEST

TUNE

SOAPscope 3.0

Try it free at www.mindreef.com

WRITTEN BY **JOHN DERRICK**

Parallel Processing for the Real-Time Enterprise

Deploying Web services on a solid platform for growth

Service orientation, Web services, self-describing data, loosely coupled applications – choose your favorite term. The enterprise IT world is moving inexorably towards architectures that will allow rapid development of applications that provide real differentiated value to their businesses.

The goal is a virtualized, real-time, extensible enterprise architecture that can quickly offer new functionality, yet integrates easily with legacy assets. This architecture must be reliable, extensible, and manageable. It must offer the highest performance for peak loads, yet not be oversized, leaving assets underutilized for typical workloads. It should offer the highest availability without duplication of expensive components.

Is it possible to build such an architecture in today's budget-constrained environments? Are standards maturing quickly enough to at least settle on a plan? Can an enterprise embark on such a quest in an evolutionary manner?

The industry answer has been "Yes, probably." Although there are many emerging standards and systems vendor strategies there is agreement in at least two areas:

1. A service-oriented architecture is a "good thing."
2. The foundation for loosely coupled applications using self-describing data is XML.

The momentum behind service orientation/Web services and XML in recent years has been stunning. Today, the majority of Fortune 1000 enterprises are utilizing XML-formatted data both for business-to-business communication and for data center applications. Zapthink, an industry analyst company with a focus on service orientation, estimates that by 2006, 25% of all LAN traffic will be XML-based, indicating a massive growth in data center XML and service orientation. It is this very growth of XML in the data center that presents an emerging concern to data center architects and those responsible for providing a high-performance infrastructure to support service orientation – how do we parse, validate, and transform all this XML data?

It's becoming clear that XML processing has a significant overhead associated with it. Large documents require manipulation at several points in the data life cycle. If done inefficiently, up to 80% of application server processing can be consumed manipulating and reformatting XML.

"The major reason for implementing an appliance approach to a given problem is that it just does a better job"

Throw Servers at It

The traditional solution to application server performance issues is to increase the amount of processing power on tap. If the server farm is already part of a tiered architecture it can be relatively easy to scale the farm by just adding another. Many of the costs in doing this, however, are hidden beyond primary hardware acquisition expenses and include:

- Server options such as memory, PCI controllers, and drives
- Network infrastructure (cables, switch ports, etc.)
- Software licenses for the application software
- Database software licenses
- Management software licenses
- Deployment and configuration time
- Test time in a simulated environment
- Implementation and benchmarking
- Ongoing management costs

These costs can quickly become prohibitive as the deployment of Web services and associated applications continues to grow. If industry estimates around the growth of data center XML LAN traffic are true, the average application server farm size will almost double by 2006. At some point, a more architecturally appropriate approach to solving this problem is required.

Upgrade to Bigger, Faster Servers

Moore's Law states that processing power effectively doubles every 18 months. Surely this provides an answer to the XML processing problem...unfortunately, it does not. While servers are indeed adopting new architectures and processors that increase the raw processing power available, it is the type of processing in addition to the amount of data center XML traffic that makes the XML problem at the same time unique and ubiquitous.

Processing of XML documents is most efficient using powerful document-

Author Bio

As CEO of Conformative Systems, John Derrick is dedicated to driving the adoption of Web services using XML and building industry-leading solutions that allow enterprises to drive down the cost of implementing high-performance XML-based applications. John has been issued 19 patents; he holds a BSEE from Iowa State University and has performed focused graduate studies at various institutions.

processing languages like XSLT. XSLT allows the programmer to parse the XML document, transform data into other formats, and perform complex business logic dependent on the nature and content of the document. The tree structure of XML means that this is best done utilizing parallel processing methodologies that can traverse many variables at the same time. At any point in the tree, other processes may be spawned to perform logic on the data or on other tree elements. Complex style sheets representing real-world business logic and data transformations can consume significant processing cycles from an application server pool. These complex, highly parallel-processing tasks cannot be performed efficiently using a general-purpose architecture.

The Appliance Approach

At some point in the technology life cycle customers look to solve problems with a dedicated, custom-built device. This is the reason every household (with owners who enjoy consuming hot bread products) has a toaster instead of making toast in the oven (a general-purpose device that also cooks roast beef and bakes cakes).

In the same way, when general-purpose servers become inefficient at solving a particular problem, server appliances become the favored approach. Routers, switches, firewalls, and load balancers are all particular instances of dedicated solutions focused on performing one task, or a range of related tasks, in the most efficient manner.

Server appliances offer a range of advantages over the equivalent general-purpose server approach.

Simple and rapid deployment

Since these appliances are dedicated to a particular task the setup, configuration, and deployment of the solution becomes more "turnkey" in nature. The solution vendor knows exactly what this device is to be used for and, as such, can develop installation and management tools that are dedicated to that purpose.

Integration into existing environments

Regardless of the type of hardware, flavor of operating system, or nature of software and tools on the system, server appliances should fit seamlessly into any given enterprise environment. Appliances should be viewed as "black box" environments that are accessible via standard interfaces (TCP/IP, SOAP, etc.) and manageable with standard tools.

The highest availability

Any dedicated server appliance should offer higher availability than its general-purpose equivalent. In a dedicated device, the usage conditions and scenarios can be predefined, preconfigured, and tested. Most server appliances offer redundant components within a single box and provide load balancing and failover for even higher availability.

'Orders of magnitude' performance gains

The major reason for implementing an appliance approach to a given problem is that it just does a better job. It's architecturally the right thing to do. In many cases, server purchases can be deferred, or existing general-purpose servers redeployed, because the appliance solution offers significant performance advantages.

XML in Hardware – Performance Gains

Deploying an XML appliance can significantly reduce the bloat of additional application servers in an enterprise, increasing application performance and reducing the cost of XML data processing by a factor of 10–15. In addition, cost savings continue throughout the life of the implementation as management, sparing, infrastructure, and availability advantages begin to accumulate. The very fact that a single two-node cluster of appliances can replace dozens of general-purpose servers makes these savings apparent.

So, how are these performance gains realized? These incredible efficiencies are delivered through the appropriate mix of hardware and software technologies specifically designed and built for the unique processing of declarative data.

The engine

These hardware solutions may be built upon custom ASICs that perform parallel processing of XML and compiled XSLT in hardware. As shown in Figure 1, the solution interfaces to the outside world via standard networking interfaces and protocols such as HTTP, TCP/IP, SOAP, etc.

The data flows through the system as follows:

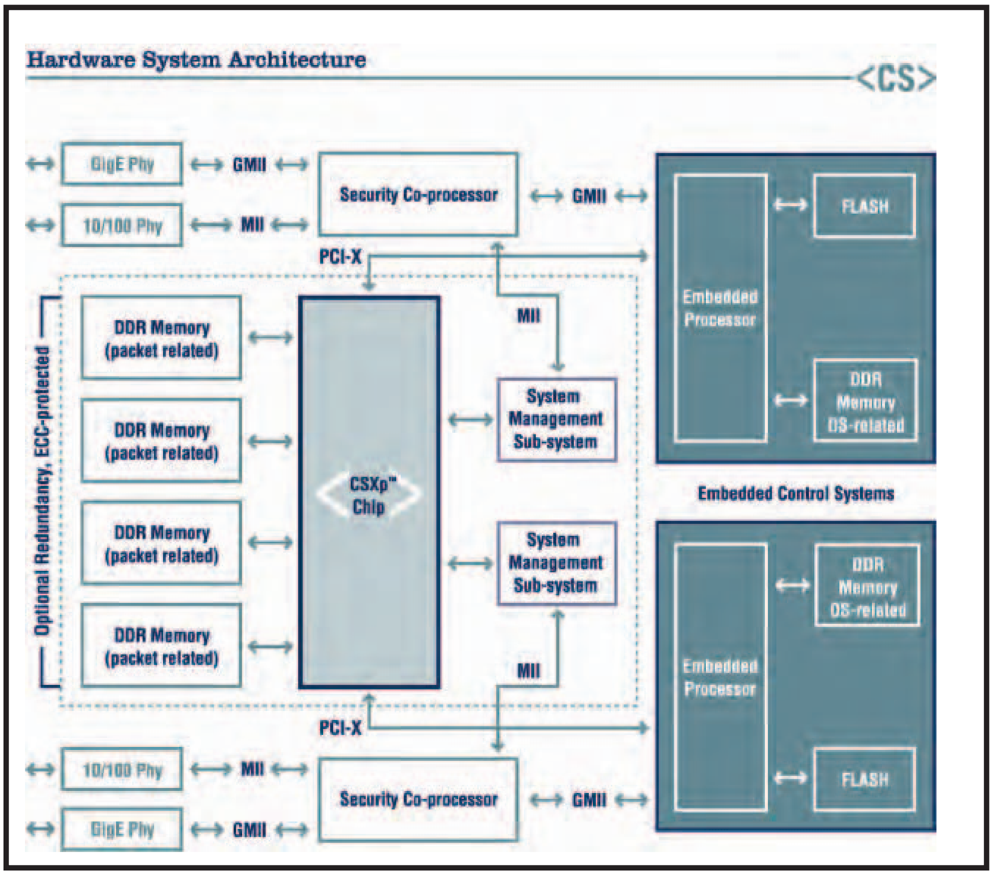


Figure 1 • Hardware systems architecture

1. XML documents are passed to the solution via one of these networking methods.
2. The document is then parsed and validated in parallel using dedicated processing hardware.
3. A custom transformation engine manages the processing of a precompiled style sheet using dedicated parallel application engines that process the document.
4. The document is then reconstituted into the output format and passed out of the solution.

Each of these functions is accelerated by the use of dedicated hardware where appropriate, providing dramatic performance advantages over traditional sequential processing methodologies.

The compiler

Any parallel processing engine has to be coupled with highly efficient compiler technologies to ensure that the efficiency of the processing engine is realized. These compiler technologies expose the available parallel processing opportunities of a declarative programming model, allowing architecturally specific hardware engines to process data for Web services with more throughput than a general-purpose processor.

In the appliance case, the compiler software is fully aware of the hardware architecture beneath it, ensuring that the maximum processing efficiencies can be gained.

Enterprise-Class Solution Components

Any solution designed for business-critical applications within the enterprise data center also demands reliability, ease of use, availability, and scalability features. These features should include:

- Software tools for rapid deployment
- Configuration tools for Web services
- Interfaces to standard SNMP-based management utilities
- Fully redundant single-server platform
- Support for failover and load balancing
- Standard software APIs for integration with Web services
- Adherence to data center industry standards

Revolution or Evolution in the Data Center?

As enterprise customers continue to search for architectures that can quickly adapt to the rapidly morphing business environment, XML (as the data standard) and Web services (as the processing standard) will become the foundation on which the majority of business logic is built.

Most enterprise customers cannot flip a switch to full service orientation. Legacy systems and applications abound in today's data center environment and will continue to remain critical to the health and success of the business. However, by basing the architecture on Web services and XML, legacy systems and data can be viewed as an asset and the organization can evolve to fully enabled service-orientated architecture in a controlled, step-wise fashion.

Web service appliances built specifically for the processing and manipulation of XML data can utilize architecturally advantaged methods to address both performance and cost challenges. This places extensible Web services and the organizations that use them on a very solid platform for growth and expansion for years to come.

JOHN.DERRICK@CONFORMATIVE.COM

For more detailed information, please visit:
lighthouseseminars.com or call **Joe Richard** at **781.821.6734**

Learn about successful implementation of content technology through the unique combination of
- Case Studies
- Project Management Techniques
- New Technology Trends

Platinum Sponsors
north plains systems

Gold Sponsors
GMC Software Technology
VIGNETTE

Analyst Sponsor
IDC Analyze the Future

Media Sponsors
cms WATCH
GILBANE REPORT
InfoWorld GET TECHNOLOGY RIGHT
WebServices Journal
XML JOURNAL

Association Sponsors
G-SAM
OSCOM OPEN SOURCE CONTENT MANAGEMENT
WoW

Register NOW!

Register now for the **CONFERENCE PLUS PACKAGE** and receive a **FREE iPod!**

LIMITED QUANTITY, REGISTER EARLY!

For details go to **lighthouseseminars.com**

The Gilbane Conference
ON CONTENT MANAGEMENT

Lighthouse Seminars

March 24-26 2004
Los Angeles, California
The Westin Bonaventure Hotel & Suites

Troubleshooting .NET Applications

Sample scenarios illustrating the use of XSLT tracing, Part 2

WRITTEN BY
KARTHIK RAVINDRAN

Instrumenting and tracing XSLT transformations by treating XSLT as a regular programming language can provide very useful diagnostic data to help analyze and troubleshoot problems.

In this article I will walk you through sample scenarios to illustrate the use of the sample XSLT tracing implementation described in Part 1 of this series [*XML-J*, Vol. 5, issue 2]. The source code and supporting files can be downloaded from www.sys-con.com/xml/sourcec.cfm.

Sample Use Case 1: Transforming Dynamically Generated XML

The scenario described in this section will use the sample XSLT tracing implementation to profile and identify performance bottlenecks in an XSLT transformation executed by applying a style sheet to XML data generated dynamically by querying a SQL Server 2000 database. The SQL Server 2000 Northwind sample database will be used in this illustration. You can read this section and walk through the Transforming Static XML use case described in the next section if you do not have access to a SQL Server 2000 installation with the Northwind sample database. The functional base and steps to configure/use the XSLT tracing feature are common to both sample scenarios.

The DynamicClient.cs file in the TracingClient project contains the client code that will be used in this scenario. Open this file and study the code and inline comments in the Main method. The scenario is very straightforward. ADO.NET code is used to fill a DataSet with data from the Orders and Order Details tables for a user-specified date range. The XML representation of the data is then extracted from the DataSet and transformed to HTML by applying a style sheet (OrderDetailsData.xsl). The style sheet is nothing fancy (kept simple to focus on the tracing functionality) and the generated HTML displays the Order data in a simple HTML table. Also note that initially the transformation is executed using a System.Xml.Xsl.XslTransform object.

Change the ADO.NET connection string in the code to point to your instance of SQL Server (you must have the Northwind sample database) and build the solution. Open the Visual Studio .NET 2003 command prompt and navigate to the TracingClient executable folder (C:\TracingDemo\TracingClient\bin\debug). The Orders and Order Details tables contain data for dates (Order date) ranging from 7/4/1996 to 5/6/1998.

Issue the following command to retrieve and transform data from the year 1996. Note the timing information reported for the phases that load the style sheet and execute the transformation. You will see that the actual transformation (note the time before and after the transform) executes in ≤ 1 second. You can open and view the HTML generated by the transformation (output.html in the same folder) in IE to see how the data is tabulated:

```
TracingClient 7/4/1996 12/31/1996
```

Now let's reexecute the process by increasing the date ranges to the following periods and observe the transformation execution times (the time before and after the transform):

1. 7/4/1996–12/31/1997
2. 7/4/1996–5/6/1998

Notice that the transformation took about 4–5 seconds for the first date range and about 8–10 seconds for the second. This clearly indicates that the style sheet is not scaling well as the size of the input XML data increases. What could be causing this drop in performance? The following are the main factors that could cause the observed drop in scalability:

1. A badly written style sheet
2. Inefficient XPath queries in an otherwise well-structured style sheet
3. Poorly structured XML data that mandates the use of inefficient XPath queries in the style sheet
4. An XSLT processor that is incapable of handling large data

Executing troubleshooting and analysis with the above factors in mind to isolate the cause of the observed behavior can be an arduous task without supporting diagnostic data. Also note that in a scenario like this the XML is generated dynamically and the performance degradation varies based on the query executed to retrieve the data. In live production environments where queries executed vary by user, such problems can occur sporadically and be tough to reproduce for subsequent analysis.

I will now show you how the sample XSLT Tracing implementation can be used to isolate the cause of the performance degradation observed in this sample scenario.

Switch to the TracingClient solution in the VS.NET 2003 IDE. Change the line of code in the Main method in DynamicClient.cs to instantiate an XsltDiagnostics .XslTransform2 object instead of a System.Xml.Xsl.XslTransform object to execute the transformation. The XsltDiagnostics assembly is referenced by the TracingClient solution. (Check this reference to make sure that it is good; you might need to refresh it if you extracted the sample files to a folder other than C:\.) Its parent namespace is imported in DynamicClient.cs. All you need to do is change this line of code from:

```
XslTransform stylesheet = new XslTransform();
```

to the following:

```
XslTransform2 stylesheet = new XslTransform2();
```

Open the application configuration file (App.config) to configure the settings required to trace the XSLT transformation. On viewing the initial XSLT tracing configuration, notice that all parameter settings are at their defaults as described in Part 1 of this series. You will need to change the following configuration settings to obtain the diagnostic data required to analyze and troubleshoot the performance degradation observed in this sample scenario:

1. Enable the option to trace XSLT transformations.
2. Enable the option to persist the source XML data (in order to obtain the dynamically generated XML to be transformed).
3. Disable the option to delete the trace logs for successful transformations (because the transformation in this scenario runs successfully to completion and the objective is to profile its performance).

After making the configuration changes specified above, your application configuration file should look like Listing 1. Note that the changes have been highlighted.

Save and rebuild the TracingClient solution. Open the VS.NET 2003 command prompt and navigate to the TracingClient executable folder (C:\TracingDemo\TracingClient\bin\debug). Issue the following command to retrieve and transform data for the period that exhibited the slowest transformation performance in the initial runs of the test:

```
TracingClient 7/4/1996 5/6/1998
```

Notice that it takes about 8–10 seconds for the transformation to complete. On completion, launch Windows Explorer and navigate to the C:\TraceOutput folder (the folder configured as the trace output path in the application configuration file). You will see the trace log (XSLTTraceOutput-<time stamp>.txt) generated for the transformation, along with an XML file containing the dynamically generated data (XMLData-<time stamp> .txt) that was transformed. The XML Data file would not have been persisted if the XSLTTracePersistSource XML option had been disabled (the default), and both files would have been deleted if the XSLTTraceDelet eLogsWhenSuccessful option had been enabled (also the default). The changes made to the default configuration settings for these options in order to enable XSLT tracing resulted in the generation of these two files. Also note that the time stamps appended to the trace log and the XML data file generated for a single transformation are the same. The time stamp can be used to match up trace logs with the corresponding XML data files in scenarios in which multiple transformations are traced during the course of an application's execution.

The instrumented style sheet (InstrumentedStyleSheet.xsl) that resulted in the generation of the trace log is persisted for reference in the TracingClient application executable folder. You can open and view this style sheet to see that the configured XSLT instructions have been instrumented. In a typical scenario in which multiple style sheets are being executed and traced, you will want to remove the line of code in the Load method of the XsltDiagnostics.XslTransform2 type that saves the instrumented style sheet to disk.

In IE, open the XML data file persisted for this transformation and view its structure. You will notice that the data is predominantly flat. Elements representing records from the Orders table are listed first, followed by elements representing records from the Order Details table.

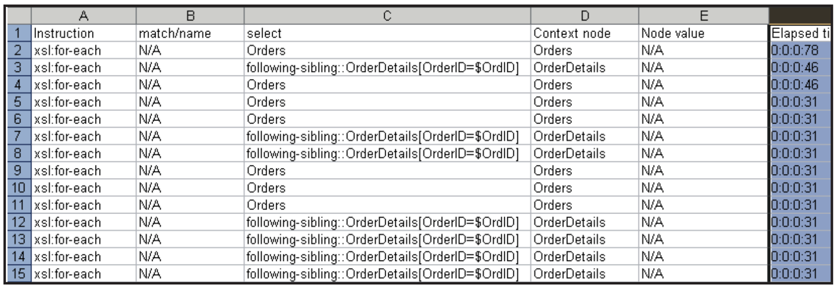
Open the trace log in Microsoft Excel (it is a tab-delimited text file and can be more easily viewed and analyzed using Excel). A spreadsheet column is generated for each of the columns in the trace log (determined by the tab delimiters). Select all the columns in the Excel spreadsheet and do a Format > Column > AutoFit selection from the Excel menu to resize them automatically to the width required to facilitate easy viewing of the data. Table 1 provides brief descriptions of the columns of data in the generated trace log and the XSLT instructions to which they apply.

Scroll down to the bottom of the spreadsheet and examine the line generated for the last instruction traced. Observe the value for this instruction in the "Cumulative elapsed time" column. This is the time taken to execute the transformation to the point of tracing the last instrumented instruction. It should be approximately equal to the transformation execution time in most cases. You will notice a line reporting the successful completion of the transformation and the total number of instructions traced during its execution (roughly 8–10 seconds, in this scenario).

Scroll back up to the top of the spreadsheet and sort the data in descending order on the "Elapsed time from last traced instruction" column. This will sort and display the trace log entries in descending order of the approximate time taken to execute each traced instruction. Figure 1 is a screen shot of the sorted trace log for an execution of this scenario on my machine.

On scrolling through the instruction trace in this sorted view, notice that the core chunk of time is spent on evaluating the select expressions of the xsl:for-each loops used to iterate the Orders and OrderDetails elements in the XML data. On examining the style sheet you will see that the xsl:for-each statement used to iterate the OrderDetails elements is nested within the xsl:for-each that iterates the Orders elements. This snippet is shown below:

```
<xsl:for-each select="Orders">
```



	A	B	C	D	E	
1	Instruction	match/name	select	Context node	Node value	Elapsed ti
2	xsl:for-each	N/A	Orders	Orders	N/A	0:0:0:78
3	xsl:for-each	N/A	following-sibling::OrderDetails[OrderID=$OrdID]	OrderDetails	N/A	0:0:0:46
4	xsl:for-each	N/A	Orders	Orders	N/A	0:0:0:46
5	xsl:for-each	N/A	Orders	Orders	N/A	0:0:0:31
6	xsl:for-each	N/A	Orders	Orders	N/A	0:0:0:31
7	xsl:for-each	N/A	following-sibling::OrderDetails[OrderID=$OrdID]	OrderDetails	N/A	0:0:0:31
8	xsl:for-each	N/A	following-sibling::OrderDetails[OrderID=$OrdID]	OrderDetails	N/A	0:0:0:31
9	xsl:for-each	N/A	Orders	Orders	N/A	0:0:0:31
10	xsl:for-each	N/A	Orders	Orders	N/A	0:0:0:31
11	xsl:for-each	N/A	Orders	Orders	N/A	0:0:0:31
12	xsl:for-each	N/A	following-sibling::OrderDetails[OrderID=$OrdID]	OrderDetails	N/A	0:0:0:31
13	xsl:for-each	N/A	following-sibling::OrderDetails[OrderID=$OrdID]	OrderDetails	N/A	0:0:0:31
14	xsl:for-each	N/A	following-sibling::OrderDetails[OrderID=$OrdID]	OrderDetails	N/A	0:0:0:31
15	xsl:for-each	N/A	following-sibling::OrderDetails[OrderID=$OrdID]	OrderDetails	N/A	0:0:0:31

Figure 1 • *A snapshot from a trace log generated by the XslTransform wrapper*

Member Name	Description	Applicable To
Instruction	XSLT instruction traced	All traced instructions
Match/name	Pattern specified for the match attribute QName specified for the name attribute	xsl:template
Select	Expression specified for the select attribute	xsl:for-each and xsl:value-of
Context node	Context node processed by the instruction when it was traced	All traced instructions
Node value	Value of context node processed by the instruction when it was traced	xsl:value-of
Elapsed time from last traced instruction	Time elapsed from the last traced instruction. This will help identify specific instructions that are slowing down a transformation	All traced instructions
Cumulative elapsed time	Cumulative time elapsed from the start (since the first instruction – the first template was traced)	All traced instructions

Table 1 • Descriptions of the trace log data and the XSLT instructions to which they apply

```
….
….
<xsl:variable name="OrdID" select="OrderID" />
….
….
<xsl:for-each select="following-
sibling::OrderDetails[OrderID=$OrdID]">
```

On examining the persisted XML that was transformed, you will again notice that the data is predominantly flat. All the Orders elements are listed first, followed by the OrderDetails elements, which results in them being sibling nodes. As each Orders element is processed, the parser must switch context from the element to its related OrderDetails elements by evaluating the predicate used in the select expression of the xsl:for-each loop to iterate the OrderDetails elements. The following-sibling XPath function is used in this sample to achieve this context switch.

An alternate option, given the structure of the data, might have been to use a select expression of ..\ OrderDetails[Order ID=$OrdID] for the inner xsl:for-each loop used to iterate the OrderDetails elements. The scalability of this expression will be the same as or, more likely, less than the following::sibling expression.

After the OrderDetails elements for an Order have been iterated, the parser's context has to switch back to the next Orders element that must be processed by the outer xsl:for-each loop. Both of these are expensive context switches whose scalability and performance will degrade in a scenario like this as the size of the XML data increases. These expensive context switches could have been avoided if the XML data had been structured to nest the OrderDetails elements within their corresponding Orders elements. This would also have resulted in the XML format being more representative of the structure of the data that it contains.

This analysis leads to the conclusion that only these two possibilities from the initial list of possible causes are applicable to the performance degradation observed in this sample scenario:

1. Inefficient XPath queries in an otherwise well-structured style sheet
2. Poorly structured XML data that mandates the use of inefficient XPath queries in the style sheet

The following steps can be executed to remedy this situation:

1. Implement code changes required to nest the OrderDetails elements within their corresponding Orders elements when generating the dynamic XML.
2. Change the select expression of the inner xsl:for-each loop to access an Order's OrderDetails elements as its direct children, thereby achieving a clean top-down processing of the data as the transformation is executed.

To achieve the nesting in the XML data in this scenario all you have to do is add the following line of code to the Main method in DynamicClient.cs to set the Nested property of the DataRelation object that links the Orders and the Order Details tables to true. This line should be added immediately after the line of code that adds the DataRelation to the DataSet object:

```
dsOrderData.Relations["OrderOrderDetails"].Nested = true;
```

The OrderDetailsData-Nested.xsl style sheet in the TracingClient application executable folder addresses the XSLT style sheet modification described above. Change the line of code in the Main method in DynamicClient.cs that loads the style sheet into the XslTransform2 instance to load this style sheet instead of OrderDetailsData.xsl.

Save the changes and rebuild the solution. Open the VS.NET 2003 command prompt and navigate to the TracingClient executable folder (C:\TracingDemo\TracingClient\bin\debug). Issue the following command to retrieve and transform the data for the period that exhibited the slowest transformation performance in the initial test runs:

```
TracingClient 7/4/1996 5/6/1998
```

Notice that the transformation – which took about 8–10 seconds initially – now completes in 1–2 seconds for the same data (in a more cleanly structured format).

Sample Use Case 2: Transforming Static XML

The scenario described in this section will use the sample XSLT tracing implementation to profile and identify performance bottlenecks in an XSLT transformation executed by applying a style sheet to static XML loaded from a hard disk. The functional base and steps to configure/use the XSLT tracing feature are the same as those described in the dynamic scenario in the previous section. Read the dynamic scenario section first (if you have not already done so) to understand these aspects.

The StaticClient.cs file in the TracingClient project contains the client code that will be used in this scenario. Open this file and study the code in the Main method (see Listing 2). You will notice that the ADO.NET code to dynamically retrieve the data to be transformed from the SQL Server Northwind sample database is not used in here. The XML data is directly loaded from the disk and transformed.

The following XML data files in the TracingClient application executable folder are provided for this scenario:

- ***OrderDetailsData.xml:*** Contains the same XML data generated

LOOK FOR YOUR FREE...

>Linux >Java >Web Services >.NET >XML >Wireless >Storage >Security
SYS-CON MEDIA
The Premier Resource for Today's Corporate & IT Decision Makers
VOL 1 ISSUE 1 SPRING 2004
IT solutions >GUIDE
WWW.SYS-CON.COM/IT
» Delivering Software as Service
» Leveraging Linux/Open Source
» Moving to a Service-Oriented Architecture
» Desktop Software: Migrating from Server to Client
» Using Developer Tools to Drive Cost Out of Software
» Application Integration
» Storage & Security
TECHNOLOGIES YOU NEED NOW!
How to Manage Your Ideas Using Today's i-Technologies
Reaching 135,000 Corporate and IT Decision Makers
$5.99US $7.99CAN

Coming this *SPRING!*

by the nonnested dynamic scenario. It contains nonnested Order data extracted from the Northwind database for the period 7/4/1996 to 5/6/1998 and can be used to re-create the performance problem illustrated in the dynamic scenario. The data in this file should be transformed using the OrderDetailsData.xsl style sheet.

- ***OrderDetailsData-Nested.xml:*** This file contains the same XML data generated by the nested dynamic scenario. It contains nested Order data extracted from the Northwind database for the period 7/4/1996 to 5/6/1998 and can be used to see how the performance problem observed in the nonnested scenario can be remedied. This data in this file should be transformed using the OrderDetailsData.xsl style sheet.

You will need to set the TracingClient project properties in the VS.NET 2003 IDE to specify TracingClient.StaticClient as the startup object, and then rebuild the solution to test this scenario. You do not have to specify a date range when executing the TracingClient application from the command prompt.

The other steps related to configuring the tracing options and analyzing the trace output are identical to the dynamic scenario. One notable difference in this scenario is that the XML data to be transformed will be loaded from the disk. As a result, you can disable the tracing configuration option (XSLTTracePersistSourceXML) to persist the XML data (the default setting).

Usage Recommendations

1. Tracing functionality of any kind will accrue performance overheads and should be used only in troubleshooting/profiling scenarios. The use of additional resources (memory, disk space, and time spent on disk I/O) cannot be avoided when there is a requirement to trace an application's execution, and XSLT tracing is no exception. Enable and use XSLT tracing only when there is a requirement to troubleshoot/profile XSLT transformations.
2. When using XSLT tracing to troubleshoot failing transformations, enable the option to delete trace logs and source XML persisted for successful transformations. This option should be enabled when there is a requirement to profile the performance of XSLT transformations that execute to completion.
3. Enable the option to persist the source XML only if the data that must be transformed is generated dynamically. There is no need to enable this option when the XML data is always loaded from disk.
4. Disable XSLT tracing as soon as the diagnostic data (trace logs and/or XML data persisted in dynamic scenarios) required to analyze and troubleshoot the problem under observation has been obtained.
5. For efficient use in automated test scenarios, envision a scenario in which the XSLT transformations executed by a .NET application must pass a suite of related automated test cases to verify successful execution. A tester can enable XSLT tracing and set the option to delete the persisted trace log and source XML files for successful transformations prior to commencing a test cycle. On completion, trace logs and source XML data files for the transformations that failed the test cycle can be found in the trace folders specified in the tracing configuration file. These can be taken offline and analyzed to determine the cause of failure and take the required corrective action. The ability to easily obtain repro data representative of the failed test cases will be a big win for testers who must verify the functionality of XSLT transformations executed by an application being tested.

Conclusion

The sample implementation discussed in this article is a proof-of-concept "scratch-the-surface" prototype that illustrates how to instrument and trace XSLT transformations and shows the value of such a feature.

I hope that I have managed to whet your appetite on this subject and get your creative juices flowing on ways to further enhance this base implementation to address additional specific requirements that you might have.

I am very interested in hearing your feedback on the value of a feature like this being built into the .NET System.Xml and MSXML stacks. The implementation of the feature if built into the .NET XML stacks could be different from the architecture discussed in this article in order to achieve a higher level of optimization and performance. However, the basic idea and concept with respect to treating XSLT as an independent programming language and generating diagnostic information that can be easily interpreted will hold. Please send your comments and feedback on this article to xmlsemfb@microsoft. com.

AUTHOR BIO

Karthik Ravindran is a technical/product lead in the Microsoft Web Data PSS group. His core responsibilities are helping enterprise customers successfully implement solutions using WebData XML technologies and working closely with the product group on delivering product feedback and inputs on the implementation of future upcoming related technologies to address top customer issues and feature requests.

This article originally appeared in .NET Developer's Journal

XMLSEMFB@MICROSOFT.COM

LISTING 1

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
<appSettings>
         <add key="TraceXSLTStyleSheets" value="true" />
         <add key="XSLTTraceOutputPath" value="c:\tra
         ceoutput" />
         <add key="TraceXSLTTemplates" value="true" />
         <add key="TraceValueOf" value="true" />
         <add key="TraceForEach" value="true" />
         <add key="XSLTTracePersistSourceXML" value="true"
         />
         <add key="XSLTTraceDeleteLogsWhenSuccessful"
         value="false" />
</appSettings>
</configuration>
```

LISTING 2

```
XslTransform2 stylesheet = new XslTransform2();
Console.WriteLine("Time before Load:" + DateTime.Now);
stylesheet.Load("OrderDetailsData.xsl");
Console.WriteLine("Time after Load:" + DateTime.Now);

XPathDocument data = new
XPathDocument("OrderDetailsData.xml");


System.IO.StreamWriter output = new
System.IO.StreamWriter("output.html");


Console.WriteLine("Time before Transform:" + DateTime.Now);
 stylesheet.Transform(data,null,output); Console.WriteLine
 ("Time after Transform:" + DateTime.Now);
```

Download the Code
www.sys-con.com/xml

FREE*CD! ($198.00 VALUE!)

Secrets of the XML Masters

Every *XML-J* Article on One CD!

XML JOURNAL
PRESENTS
XML
Resource CD
Secrets of the XML Masters
$198 VALUE FROM XML Journal
Inside: SPECIAL SUBSCRIPTION SAVINGS OFFER!
Every Issue of XML-J Ever Published on One CD!

— The Complete Works —

CD is organized into 33 chapters containing more than 450 exclusive ***XML-J*** articles!

All in an easy-to-navigate HTML format! **BONUS: Full source code included!**

ORDER AT WWW.SYS-CON.COM/FREECD

**PLUS $9.95 SHIPPING AND PROCESSING (U.S. ONLY)*

SYS-CON MEDIA

Only from the World's Leading i-Technology Publisher

XML-Journal/ Web Services Journal Readers' Choice Awards

And the winners are...

Best Book
Understanding Web Services: XML, WSDL, SOAP, and UDDI
Addison-Wesley www.aw.com
This book introduces the main ideas and concepts behind core and extended Web services technologies and provides developers with a primer for each of the major technologies that have emerged in this space. In addition, *Understanding Web Services* summarizes the major architectural approaches to Web services, examines the role of Web services within the .NET and J2EE communities; and provides information about major product offerings from BEA, HP, IBM, IONA, Microsoft, Oracle, Sun Microsystems, and others.

Best App Server for Web Services
BEA WebLogic Server
BEA Systems www.bea.com
BEA WebLogic Server 7.0 includes several significant enhancements aimed at improving the productivity of the developer. It's about fewer manual steps, cleaner code, easy packaging, and ultimately getting the job done faster, with fewer people. It simplifies deployment of massive clustered applications by providing configuration wizards and two-phase deployment capabilities.

Best GUI for Web Services
IBM WebSphere Platform
IBM www.ibm.com
IBM WebSphere is a high-performance and extremely scalable Internet infrastructure software, or middleware, for creating, running and integrating e-business applications across a variety of computing platforms. It is built on open technologies such as J2EE, XML, Eclipse, and the new Web services standards.

Best App Server for Web Services
–Winner–
BEA WebLogic Server
BEA Systems www.bea.com
–First Runner Up–
IBM WebSphere Application Server v5
IBM www.ibm.com
–Second Runner Up–
Oracle 9i Application Server
Oracle Corporation www.oracle.com
–Third Runner Up–
Sun ONE Application Server 7.0
Sun Microsystems www.sun.com

Best Book
–Winner–
Understanding Web Services: XML, WSDL, SOAP, and UDDI
Addison-Wesley www.aw.com
–First Runner Up–
XMLSPY Handbook
Wiley www.wileypub.com
–Second Runner Up–
Building Web Services with Java
Sams www.pearsoned.com
–Third Runner Up–
XML and Java: Developing Web Applications,
Second Edition www.aw.com

Best Framework for Web Services
–Winner–
EntireX 7
Software AG www.softwareag.com
–First Runner Up–
BEA WebLogic Workshop
BEA Systems www.bea.com
–Second Runner Up–
IBM WebSphere Platform
IBM www.ibm.com
–Third Runner Up–
webMethods Integration Platform
webMethods www.webmethods.com

Best GUI for Web Services Product
–Winner–
IBM WebSPhere Platform
IBM www.ibm.com
–First Runner Up–
XMLSPY *Altova* www.altova.com
–Second Runner Up–
Oracle 9i JDeveloper
Oracle Corporation www.oracle.com
–Third Runner Up–
Borland Delphi 7 Studio
Borland Software Corp. www.borland.com

Best Integrated Services Environment
–Winner–
EntireX 7
Software AG www.softwareag.com
–First Runner Up–
BEA WebLogic Platform
BEA Systems www.bea.com
–Second Runner Up–
webMethods Integration Platform
webMethods www.webmethods.com
–Third Runner Up–
IBM WebSphere Application Server v5
IBM www.ibm.com

Best Portal Platform for Web Services
–Winner–
Tamino XML Server 4.1
Software AG www.softwareag.com
–First Runner Up–
BEA WebLogic Portal
BEA Systems www.bea.com
–Second Runner Up–
IBM WebSphere Portal v4.2
IBM www.ibm.com
–Third Runner Up–
Oracle 9i AS Portal
Oracle Corporation www.oracle.com

Best Service-Oriented Architecture
–Winner–
EntireX 7
Software AG www.softwareag.com
–First Runner Up–
webMethods Integration Platform
webMethods www.webmethods.com
–Second Runner Up–
ebXML
OASIS
–Third Runner Up–
Novell exteNd
Novell, Inc. www.novell.com

Best Portal Platform for Web Services
Best Web Services Utility
Best XML Database
Tamino XML Server 4.1
Software AG www.softwareag.com
Tamino XML Server is a high-performance XML Server for reliably storing, managing, publishing, and exchanging XML documents in their native format based on open-standard Internet technologies. It is available on all major operating systems from Windows to Unix. including Linux. Tamino XML Server provides advanced high-availability features and offers many easy-to-use interfaces, tools, and functionalities that help increase developer and administrator productivity for companies of all sizes.

Best Framework for Web Services
Best Integrated Services Environment
Best Service-Oriented Architecture
Best Web Services Platform
Best Web Services Automation Tool
Best Web Services Integration Tool
Best Web Services Legacy Adapter
Best Web Services Management Tool/Platform
EntireX 7
Software AG www.softwareag.com
EntireX is Software AG's leading-edge integration server, providing an efficient way to create reusable services from existing systems, turn back-end services into Web services, and manage the growing complexity of XML-based information exchange. While many vendors view a services-oriented architecture as a new technology, EntireX was designed from the ground up to support the creation of a services view of corporate IT systems. This patented approach has been validated by years of customer success.

Best Web Service Security Solution
IBM Tivoli Access Manager
IBM www.ibm.com
IBM Tivoli Access Manager for e-business expands on IBM's open standards–based security platform by offering interoperability with third-party e-business tools. The software helps customers integrate security across e-business infrastructures, including Web services applications, allowing companies to manage security across collaborative networks that span millions of users employing Web services technologies.

Best Web Services BPM Engine
BEA WebLogic Integration
BEA Systems www.bea.com
BEA WebLogic Integration is a single solution delivering application server, application integration, business process management, and B2B integration functionality for the enterprise. Designed to speed time to value, reduce the costs of IT initiatives, and future-proof businesses. BEA WebLogic Integration relies on a standards-based, "build to integrate" approach that enables companies to rapidly develop, deploy, and integrate new Web and wireless applications, streamline complex business processes, and connect with business partners.

Best Web Services IDE
BEA WebLogic Workshop
BEA Systems www.bea.com
BEA WebLogic Workshop is an integrated development framework that empowers all application developers, not just J2EE experts, to rapidly create, test, and deploy enterprise-class Web service applications on the BEA WebLogic Platform. It provides a unified development platform that

Best Web Service Security Solution
–Winner–
IBM Tivoli Access Manager
IBM www.ibm.com
–First Runner Up–
Netegrity SiteMinder and TransactionMinder
Netegrity www.netegrity.com
–Second Runner Up–
RSA ClearTrust 5.0
RSA Security www.rsasecurity.com
–Third Runner Up–
VordelSecure
Vordel www.vordel.com

Best Web Services Platform
–Winner–
EntireX 7
Software AG www.softwareag.com
–First Runner Up–
webMethods Integration Platform
webMethods www.webmethods.com
–Second Runner Up–
XMLSPY
Altova www.altova.com
–Third Runner Up–
Novell exteNd
Novell, Inc. www.webmethods.com

Best Web Services Automation Tool
–Winner–
EntireX 7
Software AG www.softwareag.com
–First Runner Up–
IBM WebSphere Studio v5
IBM www.ibm.com
–Second Runner Up–
webMethods Integration Platform
webMethods www.webmethods.com
–Third Runner Up–
XMLSPY
Altova www.altova.com

Best Web Services BPM Engine
–Winner–
BEA WebLogic Integration
BEA Systems www.bea.com
–First Runner Up–
IBM WebSphere Busines Integration v4.2
IBM www.ibm.com
–Second Runner Up–
webMethods Integration Platform
webMethods www.webmethods.com
–Third Runner Up–
Savvion BusinessManager
Savvion, Inc. www.savvion.com

Best Web Services IDE
–Winner–
BEA WebLogic Workshop
BEA Systems www.bea.com
–First Runner Up–
IBM WebSphere Studio (Application Developer v5.0)
IBM www.ibm.com
–Second Runner Up–
Tamino Mobile Studio
Software AG www.softwareag.com
–Third Runner Up–
Oracle9i JDeveloper
Oracle Corporation www.oracle.com

Best Web Services Integration Tool
–Winner–
EntireX 7
Software AG www.softwareag.com
–First Runner Up–
BEA WebLogic Integration
BEA Systems www.bea.com
–Second Runner Up–
IBM WebSphere MQ Integrator Broker
IBM www.ibm.com
–Third Runner Up–
webMethods Integration Platform
webMethods www.webmethods.com

Best Web Services Legacy Adapter
–Winner–
EntireX 7
Software AG www.softwareag.com
–First Runner Up–
IBM WebSphere Host Integration Solution
IBM www.ibm.com
–Second Runner Up–
webMethods Integration Platform
webMethods www.webmethods.com
–Third Runner Up–
Novell exteNd Composer
Novell, Inc. www.webmethods.com

Best Web Services Management Tool/Platform
–Winner–
EntireX 7
Software AG www.softwareag.com

enables developers to easily build and connect components, data, and application business logic, while insulating them from the complexities of J2EE.

Best Web Services or XML Site

The Tamino Developer Community

Software AG www.softwareag.com

The Tamino Developer Community Web site provides useful information, software, guidance, and help to developers who intend to build XML-based software applications, solutions, or products, probably by using Software AG's Tamino XML Server. The Developer Community offers public discussion forums on a variety of topics around the Tamino XML Server.

Best Web Services Testing Tool

IBM alphaWorks Web Services Testing Area

IBM www.ibm.com

alphaWorks provides a unique opportunity for developers around the world to experience the latest innovations from IBM. These emerging "alpha code" technologies are available for download at the earliest stages of development – before they are licensed or integrated into products – allowing users to evaluate and influence IBM research and development.

Best Web Services or XML Training

Software AG's DemoZones

Software AG www.softwareag.com

Software AG established two DemoZone sites focusing on benefits that can be achieved using XML, Web services, and Software AG's products for integration and XML storage. The sites offer visitors a look into specific real-life scenarios to experience the solution space of Software AG's XML and integration technology.

Best XML Parser

IBM XML Parser for Java v4.1.2

IBM www.ibm.com

XML Parser for Java is a validating XML parser and processor written in 100% pure Java; it is a library for parsing and generating XML documents. This parser easily enables an application to read and write XML data.

Best XSLT Processor

XMLSPY

Altova www.altova.com

XMLSPY has a blazing fast XSLT processor that supports XSLT debugging and visual XSLT design.

Most Innovative Application of XML

Tamino Mobile Appllications

Software AG www.softwareag.com

Software AG's Tamino Mobile Application for Field Service Automation (FSA) provides service-oriented companies that maintain and repair their customers' technical equipment or devices with a complete, ready-to-run solution for ubiquitous access to service-relevant data. The Tamino Mobile Application for FSA provides specific services for service-order management so that starting from a service-order entry the whole service process can be managed.

–First Runner Up–
webMethods Integration Platform
webMethods www.webmethods.com

–Second Runner Up–
IBM Tivoli Monitoring for Transaction Performance
IBM www.ibm.com

–Third Runner Up–
XMLSPY
Altova www.altova.com

Best Web Services or XML Site

–Winner–
The Tamino Developer Community
Software AG www.softwareag.com

–First Runner Up–
IBM developerWorks
IBM www.ibm.com

–Second Runner Up–
XML.org
OASIS www.oasis-open.org

–Third Runner Up–
XMethods
XMethods www.xmethods.net

Best Web Services Testing Tool

–Winner–
IBM alphaWorks Web Services Testing Area
IBM www.ibm.com

–First Runner Up–
XMLSPY
Altova www.altova.com

–Second Runner Up–
iON Remote: QoS for Web Services
Santra Technology www.santra.com

–Third Runner Up–
Oracle 9i JDeveloper
Oracle Corporation www.oracle.com

Best Web Services or XML Training

–Winner–
Software AG's DemoZones
Software AG www.softwareag.com

–First Runner Up–
BEA dev2dev
BEA Systems www.bea.com

–Second Runner Up–
IBM Web Services toolkit v3.3
IBM www.ibm.com

–Third Runner Up–
Mindreef SOAPscope
Mindreef, Inc. www.mindreef.com

Best Web Services Utility

–Winner–
Tamino XML Server 4.1
Software AG www.softwareag.com

–First Runner Up–
IBM Web Services Toolkit v3.3
IBM www.ibm.com

–Second Runner Up–
XMLSPY
Altova www.altova.com

–Third Runner Up–
iON Remote: QoS for Web Services
Santra Technology www.santra.com

Best XML Database

–Winner–
Tamino XML Server 4.1
Software AG www.softwareag.com

–First Runner Up–
Oracle XML DB
Oracle Corporation www.oracle.com

–Second Runner Up–
Ipedo 3
Ipedo, Inc. www.ipedo.com

–Third Runner Up–
XML Global GoXML DB
XML Global Technologies, Inc. www.xmlglobal.com

Best XML Parser

–Winner–
IBM XML Parser for Java v4.1.2
IBM www.ibm.com

–First Runner Up–
XMLSPY
Altova www.altova.com

–Second Runner Up–
webMethods Integration Platform
webMethods www.webmethods.com

–Third Runner Up–
Oracle XDK
Oracle Corporation www.oracle.com

Best XSLT Processor

–Winner–
XMLSPY
Altova www.altova.com

–First Runner Up–
IBM LotusXSL
IBM www.ibm.com

–Second Runner Up–
Oracle XDK
Oracle Corporation www.oracle.com

–Third Runner Up–
Stylus Studio
Sonic Software www.sonicsoftware.com

Most Innovative Application of XML

–Winner–
Tamino Mobile Applications
Software AG www.softwareag.com

–First Runner Up–
IBM WebSphere Portal v 4.2
IBM www.ibm.com

–Second Runner Up–
XMLSPY
Altova www.altova.com

–Third Runner Up–
ColdFusion MX
Macromedia www.macromedia.com

Now

More

Than

Ever

Before...

JAVA DEVELOPER'S JOURNAL

JDJ

Means

Business

EXCLUSIVE: WHERE IS TECH GOING IN 2004?
JAVA DEVELOPER'S JOURNAL
JDJ
JANUARY 2004
VOLUME:9 ISSUE:1
SPECIAL REPORT:
SCOTT McNEALY
2004 PREDICTIONS
PLUS...
Viewpoint: Deutsch's Fallacies, 10 Years After
Feature: Multiplatform Frames
Data Integration: Which EII Solution Is Right for You?
TAKE A RIDE WITH THE
CEO OF BEA
EDGE 2004 EAST
International Conference & Expo
BOSTON, MA
THE WORLD'S LEADING i-TECHNOLOGY MAGAZINE WWW.JAVADEVELOPERSJOURNAL.COM

Subscribe Now!

$69^99 1 YEAR/ 12 ISSUES

www.SYS-CON.com/JDJ

or call

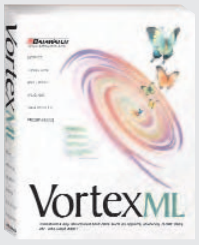
1-888-303-5282

The World's Leading *i*-Technology Magazine Just Got Better!

Now with expanded business coverage, *JDJ* is the world's premier independent, vendor-neutral print resource for the ever-expanding international community of IT professionals.

The Most Popular *i*-Technology Magazine in the World !

SYS-CON MEDIA
The World's Leading i-Technology Publisher



REVIEWED BY **KETAN PATEL**

VORTEXML DESIGNER 3.0

BY DATAWATCH CORPORATION

DATAWATCH CORPORATION

175 Cabot Street, Suite 503
Lowell, MA 01854-3633

Phone: 800 445-3311

E-Mail: sales@datawatch.com

Web: www.datawatch.com

TEST PLATFORM

Dell Dimension 4550; Intel Pentium 4 2.4 GHz; Windows XP Pro w/SP1; NTFS; 640 MB RAM; 120 GB EIDE Drive; 30 MB used for application.

PRODUCT REQUIREMENTS

Pentium III or higher; Windows 98, ME, NT4.0, 2000, or XP; Internet Explorer 5.5 SP2 or higher; 64 MB RAM; 30 MB disk space.

PLATFORMS

Windows 98, ME, NT4.0, 2000, or XP

Pricing: $599

Datawatch recently released version 3.0 of VorteXML Designer. In addition to schema support, this new version includes several updates and features that further simplify the process of generating XML by enriching any text document.

Overview

In my consulting practice, most of the solutions I recommend and build for clients require contextual variations that justify the use of XML. The ability to have attributes and metadata associated with specific documents becomes critical to processing instance-specific variations. While the actual solutions are straightforward, the challenges often come down to the laborious and tedious task of understanding the variances and populating these exceptions into the XML from the existing systems.

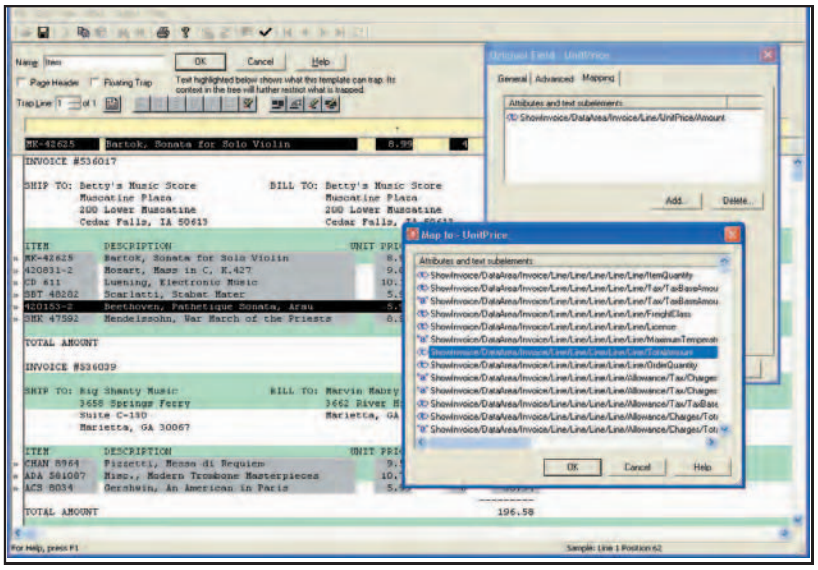
VorteXML Designer from Datawatch Corporation has provided me with an easy-to-use tool to address this need. VorteXML allows me to build and test profiles through a visual interface that extracts, transforms, and maps data from existing text documents into XML. Version 3.0 of VorteXML provides support for W3C Schema. While I was expecting the basic level of support that made the mapping to XSD possible, Datawatch has done a good job of integrating schema specifications with the mapping capabilities of the parsing and recognition engine. As a result, version 3.0 of VorteXML has many enhancements that make the use of XML content standards, including ebXML, OAGIS, and HL7, dramatically easier to implement. The process of mapping a text invoice to an OAGIS ShowInvoice Business Object Document took less than one hour (see Figure 1).

While the enhancements for data mapping and transformation are excellent, the continued absence of basic XSL management and presentation capabilities create a need for additional tools and software for building solutions that require presentation.

XML Schema Capabilities

Version 3.0 adds the ability to generate an XSD Schema from the input data, mappings, and other metadata. This capability allows for deriving custom schemas for the internal structure of the source data, or using one of the industry-standard schemas and then cutting it down to size to suit your needs. Many of the industry-standard schemas have extensions and optional data that are required of any comprehensive standard but are not necessary in most specific instances of their use.

As the schema is loaded, the elements and attributes are assigned the supported base data types from the schema. These are optional and can be dismissed or edited. VorteXML uses the base data types to map the input data to the XSD data types. VorteXML also uses



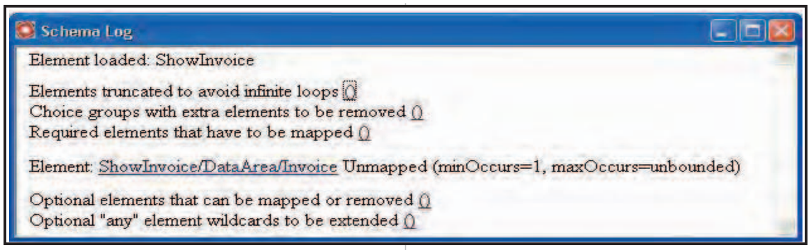
Text highlighted below shows what this template can trap. Its context in the tree will further restrict what is trapped.

```
MK-42625     Bartok, Sonata for Solo Violin              8.99     4
INVOICE #536017

SHIP TO: Betty's Music Store              BILL TO: Betty's Music Store
         Muscatine Plaza                           Muscatine Plaza
         200 Lower Muscatine                       200 Lower Muscatine
         Cedar Falls, IA 50613                     Cedar Falls, IA 50613

ITEM         DESCRIPTION                              UNIT PRI
MK-42625     Bartok, Sonata for Solo Violin              8.
420831-2     Mozart, Mass in C, K.427                    9.
CD 611       Luening, Electronic Music                  10.
SBT 48282    Scarlatti, Stabat Mater                     5.
420153-2     Beethoven, Pathetique Sonata, Arau          5.
SMK 47592    Mendelssohn, War March of the Priests       8.

TOTAL AMOUNT

INVOICE #536039

SHIP TO: Big Shanty Music                 BILL TO: Marvin Mabry
         3658 Springs Ferry                        3662 River H
         Suite C-130                               Marietta, GA
         Marietta, GA 30067

ITEM         DESCRIPTION                              UNIT PRI
CHAN 8964    Pizzetti, Messa di Requiem                  9.
ADA 581087   Misc., Modern Trombone Masterpieces        10.
ACS 8034     Gershwin, An American in Paris              5.

TOTAL AMOUNT                                          196.58
```

Attributes and text subelements
ShowInvoice/DataArea/Invoice/Line/UnitPrice/Amount

Attributes and text subelements
ShowInvoice/DataArea/Invoice/Line/Line/Line/Line/Line/ItemQuantity
ShowInvoice/DataArea/Invoice/Line/Line/Line/Line/Line/Tax/TaxBaseAmou
ShowInvoice/DataArea/Invoice/Line/Line/Line/Line/Line/Tax/TaxBaseAmou
ShowInvoice/DataArea/Invoice/Line/Line/Line/Line/Line/FreightClass
ShowInvoice/DataArea/Invoice/Line/Line/Line/Line/Line/License
ShowInvoice/DataArea/Invoice/Line/Line/Line/Line/Line/MaximumTemperat
ShowInvoice/DataArea/Invoice/Line/Line/Line/Line/Line/TotalAmount
ShowInvoice/DataArea/Invoice/Line/Line/Line/Line/Line/OrderQuantity
ShowInvoice/DataArea/Invoice/Line/Line/Line/Line/Allowance/Tax/Charges
ShowInvoice/DataArea/Invoice/Line/Line/Line/Line/Allowance/Tax/Charges
ShowInvoice/DataArea/Invoice/Line/Line/Line/Line/Allowance/Tax/TaxBase
ShowInvoice/DataArea/Invoice/Line/Line/Line/Line/Allowance/Charges/Tot
ShowInvoice/DataArea/Invoice/Line/Line/Line/Line/Allowance/Charges/Tot

Figure 1 • Mapping extracted fields to show elements the results of its parsing capabilities to minimize the number of required elements that are used in generating the output XML. This allows users to minimize the number of element and attribute mappings required to create a valid XML document. Any missed or unmapped elements are shown in the schema log (see Figure 2).



```
Schema Log
Element loaded: ShowInvoice

Elements truncated to avoid infinite loops (0)
Choice groups with extra elements to be removed (0)
Required elements that have to be mapped (0)

Element: ShowInvoice/DataArea/Invoice Unmapped (minOccurs=1, maxOccurs=unbounded)

Optional elements that can be mapped or removed (0)
Optional "any" element wildcards to be extended (0)
```

Figure 2 • Dynamic log identifies elements required for validation

Usability Features

VorteXML 3.0 has made several usability improvements, including enhanced trapping and extraction capabilities, that allow for the assignment of attributes based on standard convention. In addition, Datawatch added functionality that allows for the same level of scrutiny on the output side as they had on the input side (see Figure 3). Other enhancements include:

- New formula editor
- Improved analysis window
- Improved verify function

While all of these features further reduce the learning curve for users new to XML, those familiar with XML, and the specific syntax they are using, may find it easier to make repetitive changes (changing the attribute constraints for all text elements) in an external editor manipulating the XML directly.

Summary

More and more solutions require the use of XML because of its ability to tag the required metadata for use between two systems. Using schema with its data types is a necessary evolution of this process and is essential to validation and integrity of the XML instance. In many cases, the source data consists of HTML, reports, log files, or invoices, which only exist as text files that require conversion to XML.

With the addition of XSD schema support and capabilities for outputting schema definitions, VorteXML has become an essential tool for consultants and developers of solutions involving XML. The usability features that made the previous version valuable for extraction, parsing, and transformation of existing documents are now matched with similar capabilities on the XML side.

Author Bio

Ketan Patel serves as principal founder for Synapsys Technology Corporation. While at Synapsys, He has invented a behavior-based consulting methodology and a minimally invasive approach to building solutions and has successfully brought information-based products to market in the telecommunications, healthcare, and supply-chain industries.

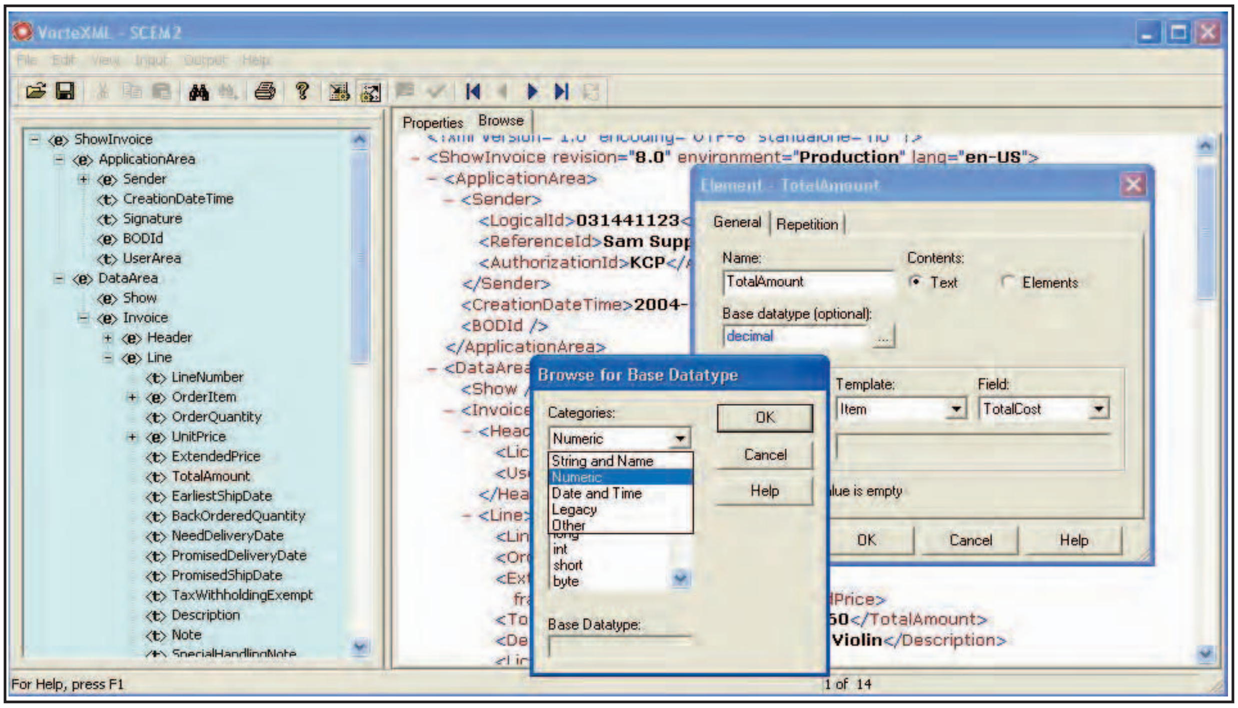
Product Snapshot

Level: Beginner to advanced programmers

Target Audience: Developers and consultants using XML for:
- Web services
- Implementation of XML standards for B2B
- Web enablement of legacy applications
- Data collection for business intelligence

Pros:
- Superior XSD Schema support
- Requires no programming skills
- Ability to localize namespaces and external references
- Reduces XML output size by removing optional data elements
- Very fast performance for XML Schema transformations

Cons:
- VorteXML Designer available for Windows only
- No XSL presentation capabilities
- Limited advanced XML manipulation capabilities



```
- <e> ShowInvoice
  - <e> ApplicationArea
    + <e> Sender
      <t> CreationDateTime
      <t> Signature
      <e> BODId
      <t> UserArea
  - <e> DataArea
      <e> Show
    - <e> Invoice
      + <e> Header
      - <e> Line
          <t> LineNumber
        + <e> OrderItem
          <t> OrderQuantity
        + <e> UnitPrice
          <t> ExtendedPrice
          <t> TotalAmount
          <t> EarliestShipDate
          <t> BackOrderedQuantity
          <t> NeedDeliveryDate
          <t> PromisedDeliveryDate
          <t> PromisedShipDate
          <t> TaxWithholdingExempt
          <t> Description
          <t> Note
          <t> SpecialHandlingNote
```

```
- <ShowInvoice revision="8.0" environment="Production" lang="en-US">
  - <ApplicationArea>
    - <Sender>
        <LogicalId>031441123<
        <ReferenceId>Sam Supp
        <AuthorizationId>KCP</
      </Sender>
      <CreationDateTime>2004-
      <BODId />
    </ApplicationArea>
  - <DataArea
      <Show /
    - <Invoice
      - <Head
          <Lic
          <Us
        </Hea
      - <Line>
          <Lin
          <Or
          <Ext
            fr                    dPrice>
          <To                    50</TotalAmount>
          <De                    Violin</Description>
```

Name: TotalAmount
Contents: (•) Text ( ) Elements
Base datatype (optional): decimal
Template: Item
Field: TotalCost
lue is empty

Categories: Numeric
String and Name
Numeric
Date and Time
Legacy
Other
int
short
byte
Base Datatype:

Figure 2 • 'Unmapped' element attribute allows dramatic reduction of final XML sizes

WRITTEN BY **AJAY VOHRA & DEEPAK VOHRA**

DTD and XML Schema Structures

Conversion methods for J2EE and XML developers

This article compares Document Type Definition (DTD) and XML Schema elements. Java 2 Enterprise Edition (J2EE) developers use DTDs and schemas in J2EE/XML applications. When a DTD for an XML document is provided and validation with an XML Schema is required, the DTD-to-XML Schema conversion creates an XML Schema document corresponding to the DTD document.

Overview

A DTD defines the structure of an XML document and defines a document's element types, subelement types, and the order and number of each element type. It also declares the attributes, entities, notations, processing instructions, and comments. An XML Schema is an XML-based representation of the structure of a XML document. Among the advantages of XML Schema is support for data types and namespaces. XML Schema, being XML based, is an extensible, complex representation of an XML document structure.

In this tutorial the different DTD declarations and their equivalent declarations in an XML Schema, and the equivalent of the XML document DOCTYPE declaration, are discussed, including:

- DOCTYPE
- Comment
- Processing instructions
- Element
- Attribute
- Entity
- Notation

DOCTYPE Declaration

The DOCTYPE declaration in an XML document refers to a public URI and the system URI of an external DTD. If the XML document is validated, the document elements and attributes are validated with the element and attribute definitions specified in the DOCTYPE DTD. A DTD specified with a system and public identifiers is an external subset DTD. A DTD may also be specified as an internal subset. An internal subset DTD is specified in the DOCTYPE declaration, which may consist of a combination of internal and external DTDs.

The DOCTYPE declaration in an XML document is specified with the <!DOCTYPE> declaration.

```
<!DOCTYPE  root_element  ((SYSTEM
"system_uri")|(PUBLIC "public_uri"
"system_uri")>
```

The system URI of a schema document is specified in an XML document with the xsi:schemaLocation and xsi:noNamespaceSchemaLocation attributes in the root element. The xsi:schemaLocation is used to specify a namespace schema. The xsi:noNamespaceSchemaLocation is used to specify a schema without a namespace.

```
<root_element
xmlns:xsi=http://www.w3.org/2001/
  XMLSchema-instance
xsi:noNamespaceSchemaLocation=
  "system_uri"
  xsi:schemaLocation=
    "namespace_uri  system_uri ">
```

Comment Declaration

Comments in an XML document or a DTD are not displayed in the output document. They may be placed only after the XML declaration. They may not be placed within a tag, but may be placed within a DOCTYPE declaration.

In a DTD comments are specified with a comment declaration.

```
<!-- DTD Comment-->
```

In a schema comments are declared in the <xs:documentation/> element.

```
<xs:annotation>
 <xs:documentation>Schema Comment
  </xs:documentation>
     </xs:annotation>
```

Processing Instructions Declaration

Processing instructions are used to include information in a DTD to be processed by applications. The XML declaration is an example of a processing instruction. Processing instructions in a DTD are specified with a processing instruction declaration.

```
<? PITarget Instruction?>
```

The PITarget "xml" or "XML" may only be used in an XML declaration. In a schema the <xs:appInfo/> element specifies the instruction for an application.

```
<xs:annotation>
  <xs:appinfo>
    <fn:application
     Instruction>
         Instruction for Application
       </fn:application
     Instruction>
  </xs:appinfo>
</xs:annotation>
```

Element Declaration

Element declarations in a DTD specify the type and number of elements that may be included in an XML document. The element declaration also specifies the subelements and the order of the subelements.

Elements in a DTD are represented with <!ELEMENT> declarations.

```
 <!ELEMENT A (B, C)>
```

AUTHOR BIOS

Ajay Vohra is a senior software engineer with Compuware.

Deepak Vohra is a Web developer and a NuBean consultant.

Element A has subelements B and C. Element declarations in an XML Schema are represented with the <xsd:element/>.

```
<xsd:element  name="element_name"
 type="element_type" />
```

An <xs:element/> declaration in a schema may be in a <xs:schema/>, <xs:sequence/>, <xs:choice/>, or a<xs:all/> element. The following sections discuss different types of element declarations.

Text element

A text element may consist only of text. A text element is declared in a DTD by specifying the element as a parsed character data (PCDATA) element.

```
<!ELEMENT A (#PCDATA)>
```

In an XML Schema a text element is declared by specifying the type attribute of the element as xs:string.

```
<xs:element name="A"
type="xs:string"/>
```

A PCDATA element may also be declared in a schema with a <xs:complexType/>.

```
<xsd:element name="A">
    <xsd:complexType mixed="true">
      <xsd:complexContent>
        <xsd:restriction
         base="xsd:anyType">
        </xsd:restriction>
     </xsd:complexContent>
   </xsd:complexType>
 </xsd:element>
```

Empty element

An empty element does not consist of any subelements or text. An empty element is declared in a DTD by specifying the element as an EMPTY element.

```
<!ELEMENT A EMPTY>
```

In a schema an empty element is represented by setting attribute “mixed” of element <xs:complexType> to “false.”

```
  <xsd:element name="A">
    <xsd:complexType mixed="false">
      <xsd:complexContent>
        <xsd:restriction
         base="xsd:anyType">
        </xsd:restriction>
      </xsd:complexContent>
    </xsd:complexType>
   </xsd:element>
```

Any element

An element may be specified with variable text or subelements within the element. A variable element is declared in a DTD by specifying the element as an ANY element.

```
<!ELEMENT A ANY>
```

Element A may consist of variable text or subelements. The schema representation of a variable element is specified with the <xs:any/> element.

```
  <xs:element name="A">
    <xs:complexType>
      <xs:sequence>
<xs:any minOccurs="0"
 maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Element sequence

An element in an XML document may consist of a sequence of subelements. An element sequence is specified in a DTD with a list of subelements.

```
<!ELEMENT A(B, C, D)>
```

Elements B, C, and D are a sequence of elements in Element A. In a schema a sequence of elements is specified with <xs:sequence/> .

```
    <xs:element name="A" >
      <xs:complexType>
        <xs:sequence>
             <xs:element name="B"
               type="xs:string"/>
             <xs:element name="C"
              type="xs:string"/>
              <xs:element name="D"
              type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
       </xs:element>
```

Element choice

An element in an XML document may consist of a choice of elements. An element choice is specified in a DTD with a choice (|) of subelements.

```
<!ELEMENT A (B|C|D)>
```

Elements B, C, and D are a choice of elements in Element A. In a schema a choice of elements is specified with <xs:choice/>.

```
<xs:element name="A" >
     <xs:complexType>
         <xs:choice>
         <xs:element name="B"
            type="xs:string"/>
          <xs:element name="C"
           type="xs:string"/>
         <xs:element name="D"
          type="xs:string"/>
      </xs:choice>
     </xs:complexType>
    </xs:element>
```

Element with sequence and choice

An element in an XML document may consist of a sequence and a choice of subelements.

An element sequence and choice is specified in a DTD with a combination of a list and choice (|) declarations.

```
<!ELEMENT A ((B,C)|D>
```

In a schema an element consisting of a sequence and a choice is declared with a combination of <xs:sequence/> and <xs:choice/> elements.

```
<xs:element name="A" >
   <xs:complexType>
     <xs:choice>
       <xs:sequence>
          <xs:element name="B"
           type="xs:string"/>
             <xs:element name="C"
              type="xs:string"/>
         </xs:sequence>
       <xs:element name="D"
        type="xs:string"/>
      </xs:choice>
     </xs:complexType>
 </xs:element>
```

Element with cardinality

The cardinality of an element is the number of occurences of that element. In a DTD a cardinality of 0 or 1 is represented with “?”; a cardinality of 0 or more is represented with “*”; and a cardinality of 1 or more is represented with “+”. An element or an element group may have a cardinality. An element with subelements with a cardinality may be specified in a DTD with the cardinality notation (?, *, +).

```
<!ELEMENT  A (B?, (C*, D+)*)>
```

In a schema, cardinality is represented with the minOccurs and maxOccurs attributes (see Listing 1).

The minOccurs and maxOccurs attributes may be declared in the <xs:element/>, <xs:all/>, <xs:choice/>, <xs:sequence/>, and <xs:any/> elements.

Element with PCDATA and subelements

An element in an XML document may consist of both text and subelements. An element with text and subelements is declared in a DTD by specifying the element as a list of subelements consisting of a PCDATA subelement.

```
<!ELEMENT  A(#PCDATA, B, C)>
```

An element with text and subelements in a schema is specified with the <xs:complexType/> attribute "mixed" set to "true".

```
<xs:element name="A" >
  <xs:complexType mixed="true">
    <xs:sequence>
         <xs:element name="B"
          type="xs:string"/>
           <xs:element name="C"
            type="xs:string" />
          </xs:sequence>
         </xs:complexType>
       </xs:element>
```

Attribute Declaration

Attributes are additional information associated with XML document elements. An attribute in a DTD is defined with an <!ATTLIST> declaration.

```
<!ATTLIST Element_Name (Attribute_
 Name attribute_type  default_value)*>
```

An ATTLIST may define multiple attributes. The attribute_type may be "CDATA" enumerations of values "NOTATION", "ID", "IDREF", "IDREFS", "ENTITY", "ENTITIES", "NMTOKEN", or "NMTOKENS". The default_type may be "#REQUIRED", "#IMPLIED", or "#FIXED AttValue".

In a schema an attribute is declared with the <xs:attribute/> element.

```
<xs:attribute default ="default_
  value" fixed ="fixed_value" name
  ="attribute_name" use = (optional |
  required) :        optional/>
```

The <xs:attribute/> declaration may be in an <xs:schema/>, <xs:complexType/>, <xs:restriction/>, <xs:extension/>, or <xs:attributeGroup/> element.

CDATA attributes

Character data (CDATA) attributes are text attributes. Text attributes in a DTD are declared by specifying an attribute as a CDATA attribute.

```
<!ATTLIST  A
       a  CDATA #REQUIRED
       b  CDATA #IMPLIED
       c  CDATA #FIXED "AttValue">
```

Element A has attributes a, b, and c.

In a schema a CDATA attribute is specified with the attribute type xs:string.

```
<xs:element name="A" >
   <xs:complexType>
         <xs:attribute name="a"
          type="xs:string"
          use="required"/>
          <xs:attribute name="b"
           type="xs:string"
           use="optional"/>
               <xs:attribute name="c"
type="xs:string"
 fixed="AttValue"/>
       </xs:complexType>
     </xs:element>
```

The attributes may be included in an element as a <xs:attributeGroup/>. The <xs:attributeGroup/> element declares a group of attributes (see Listing 2).

Enumerated Attributes

Enumerated attributes specify a choice of attribute values. In a DTD an enumerated attribute is declared with a choice(|) of attribute values.

```
<!ATTLIST A  a(x|y|z)>
```

In a schema an enumerated attribute is declared with the <xs:enumeration/> element (see Listing 3).

Data types attributes

An attribute in a DTD with data type "NOTATION", "ID", "IDREF", "IDREFS", "ENTITY," "ENTITIES", "NMTOKEN", or "NMTOKENS" is specified as in the following DTD declaration.

```
<!ATTLIST A
      a NOTATION
      b ID
      c IDREF
      d IDREFS
      e ENTITY
      f ENTITIES
      g NMTOKEN
      h NMTOKENS>
```

LISTING 1 •

```
<xs:element name="A" >
      <xs:complexType>
        <xs:sequence>
          <xs:element name="B" type="xs:string" minOc
           curs="0"  maxOccurs="1" />
          <xs:choice minOccurs="0"  maxOccurs="unbounded">
            <xs:element name="C" type="xs:string"
             minOccurs="0"
             maxOccurs="unbounded"/>
            <xs:element name="D" type="xs:string"
              minOccurs="1"
             maxOccurs="unbounded"/>
          </xs:choice>
       </xs:sequence>
     </xs:complexType>
   </xs:element>
```

LISTING 2 •

```
<xs:element name="A" >
  <xs:complexType>
    <xs:attributeGroup ref="AttrGroup"/>
  </xs:complexType>
 </xs:element>
  <xs:attributeGroup  name="AttrGroup">
   <xs:attribute name="a"  type="xs:string" use="required"/>
    <xs:attribute name="b" type="xs:string" use="optional"/>
    <xs:attribute name="c"  type="xs:string"
     fixed="AttValue"/>
   </xs:attributeGroup>
```

LISTING 3 •

```
<xs:element name="A" >
  <xs:complexType>
    <xs:attribute name="a">
      <xs:simpleType>
        <xs:restriction base="xs:string">
           <xs:enumeration value="x"/>
           <xs:enumeration value="y"/>
           <xs:enumeration value="z"/>
         </xs:restriction>
       </xs:simpleType>
     </xs:attribute>
   </xs:complexType>
 </xs:element>
```

LISTING 4 •

```
<xs:element name="A" >
   <xs:complexType>
     <xs:attribute name="a"  type="xs:NOTATION"/>
     <xs:attribute name="b"  type="xs:ID"/>
     <xs:attribute name="c"  type="xs:IDREF"/>
     <xs:attribute name="d"  type="xs:IDREFS"/>
     <xs:attribute name="e"  type="xs:ENTITY"/>
     <xs:attribute name="f"  type="xs:ENTITIES"/>
     <xs:attribute name="g"  type="xs:NMTOKEN"/>
     <xs:attribute name="h"  type="xs:NMTOKENS"/>
 </xs:complexType>
</xs:element>
```

Download the Code
www.sys-con.com/xml

In a schema an element with "NOTATION", "ID", "IDREF", "IDREFS", "ENTITY", "ENTITIES", "NMTOKEN", or "NMTOKENS" data type attributes is specified by setting the "type" attribute of the <xs:attribute/> element (see Listing 4).

Entity Declaration

Entities reference data that is abbreviated or that may be found at an external location. In a DTD an entity may be a general entity or a parameter entity. A general entity is represented in a DTD with an <!ENTITY> declaration.

```
<!ENTITY Entity_name
Entity_value|((SYSTEM System_uri |
PUBLIC Public_uri System_uri) NDATA
Notation_name)>
```

A parameter entity is represented with a <!ENTITY> declaration.

```
<!ENTITY Entity_name
Entity_value|(SYSTEM System_uri |
PUBLIC Public_uri System_uri)>
```

System_uri is entity's system identifier and Public_uri is entity's public identifier.

XML Schema does not have an equivalent of an ENTITY declaration. In a schema an entity is specified with the <xs:simpleType/> element. An entity is represented in a DTD by the following ENTITY declaration:

```
<!ENTITY % complexDerivationSet
"CDATA">
```

which is represented in a schema with the following <xs;simpleType/> declaration:

```
<xs:simpleType  name="complexDeriva-
tionSet">
              <xs:restriction
base="xs:string"/>
               </xs:simpleType>
```

Notation Declaration

Notations are used to identify the format of unparsed entities, elements with notation attributes, or processing instructions. In a DTD a notation is specified with a <!NOTATION> declaration.

```
<!NOTATION  notation_name (ExternalID
| PublicID)>
```

In a schema a notation is specified with the <xs:notation/> element.

```
<xs:notation name ="notation_name"
public ="PublicID" system ="Exter-
nalID"/>
```

A notation is declared in a DTD with the following declaration:

```
<!NOTATION gif  PUBLIC  "image/gif">
```

and represented in a schema with the following declaration:

```
<xs:notation name="gif"
public="image/gif">
```

Conclusion

Based on the DTD declarations and the equivalent XML Schema declarations an XML/J2EE developer should be able to convert a DTD to an XML Schema, and an XML Schema to a DTD document.

AJAY_VOHRA@YAHOO.COM
DVOHRA09@YAHOO.COM

WRITTEN BY **DAVID S. LINTHICUM**

Application Integration: Addressing the Issues

One-stop shopping is not a reality

Application integration brings a combination of problems. Each organization and trading community has its own set of integration issues that must be addressed. Because of this, it is next to impossible to find a single technological solution set and/or standard that can be applied universally. Therefore, each application integration solution will generally require different approaches. At this time, and in the foreseeable future, one-stop shopping is simply not an application integration reality.

Although approaches to application integration vary considerably, it is possible to create some general categories. These approaches include:

- Information-oriented
- Process integration–oriented
- Service-oriented
- Portal-oriented

In this column I review the differences and the enabling XML-based standards that are applicable.

Information-Oriented Approach

Technologists who promote the information-oriented approach to application integration argue that integration should occur between the databases (or proprietary APIs that produce information, such as BAPI) – that is, databases or information-producing APIs should be viewed as the primary points of integration. Within information-oriented application integration, there are many approaches. Information-oriented solutions can be grouped into three categories: data replication, data federation, and interface processing.

Data replication is simply moving data between two or more databases. These databases can come from the same vendor or from many vendors. They can even be databases that employ different models. The fundamental requirement of database replication is that it accounts for the differences between database models and database schemas by providing the infrastructure to exchange data. Solutions that provide for such infrastructures are plentiful and inexpensive, and many leverage common XML as the canonical format, accounting for the differences in schema using XSLT.

"A common set of methods among enterprise applications invites reusability"

Now known as EII (enterprise information integration), database federation is the integration of multiple databases and database models into a single, unified view of the databases. Put another way, database federations are virtual enterprise databases that comprise many real, physical databases. Although database federation has been around for some time, the solution set has been perfected only recently.

Database federation software places a layer of software (middleware) between the physical distributed databases and the applications that view the data. This layer connects to the back-end databases using available interfaces and maps the physical databases to a virtual database model that exists only in the software. The application uses this virtual database to access the required information. The database federation handles the collection and distribution of the data as needed to the physical databases.

Interface-processing solutions use well-defined application interfaces to focus on the integration of both packaged and custom applications. Currently, interest in integrating popular ERP applications (e.g., SAP, PeopleSoft, and Oracle) has made this the most exciting application integration sector. These interfaces, while producing mostly data, also have the ability to act as an application service provider as well, and in many cases produce XML and leverage Web services interfaces. SAP, Oracle Financials, and PeopleSoft are examples of application interfaces that leverage XML standards in one shape or form.

Process Integration–Oriented Approach

Simply put, process integration–oriented products layer a set of easily defined and centrally managed processes on top of existing sets of processes contained within a set of enterprise applications. In the world of Web services we are calling this orchestration, leveraging standards such as BPEL. The concepts are almost exactly the same.

Process integration is the science and mechanism of managing the movement of data, and the invocation of services in the correct and proper order to support the management and execution of common processes that exist in and between applications. Process integration-oriented application integration

Author Bio

David S. Linthicum is the former CTO of Mercator, and author of the ground-breaking book Enterprise Application Integration. *His latest book is* Next Generation Application Integration.

provides another layer of easily defined and centrally managed processes that exist on top of an existing set of services and data contained within a set of applications.

The goal is to bring together relevant processes found in an enterprise or trading community to obtain the maximum amount of value while supporting the information flow and control logic between these processes. These products view the middleware, or the plumbing, as a commodity and provide easy-to-use visual interfaces for binding these processes together.

In reality, business process integration is another layer of value resting upon existing application integration solutions, solutions that include integration servers, application servers, distributed objects (such as Web services), and other middleware layers. Process integration offers a mechanism to bind disparate processes together, and to create process-to-process solutions that automate tasks once performed manually.

Service-Oriented Approach

Service-oriented application integration allows applications to share common business logic, or methods. This is accomplished either by defining methods that can be shared, and therefore integrated, or by providing the infrastructure for such method sharing such as Web services. Methods may be shared either by being hosted on a central server, by accessing them inter-application (e.g., distributed objects), or through standard Web services mechanisms such as .NET.

Attempts to share common processes have a long history, one that began more than 10 years ago with the multi-tiered client/server – a set of shared services on a common server that provided the enterprise with the infrastructure for reuse, and now, for integration – and the distributed object movement. Reusability is a valuable objective. A common set of methods among enterprise applications invites reusability and, as a result, significantly reduces the need for redundant methods and/or applications.

While most methods exist for single-organization use, we are learning that there are times when it makes sense to share between organizations. In a new twist on the longstanding practice of reusability, we are now hoping to expand this sharing beyond intra-enterprise to trading partners as well. For example, sharing a common logic to process credit requests from customers, or to calculate shipping costs using a set of Web services.

Portal-Oriented Approach

Portal-oriented B2B application integration allows us to view a multitude of systems – both internal enterprise systems and external trading community systems – through a single user interface or application. Portal-oriented application integration benefits us by eliminating the back-end integration problem altogether by extending the user interface of each system to a common user interface (aggregated user interface) – most often a Web browser. As a result, it integrates all participating systems through the browser, although the applications are not directly integrated within or between the enterprises.

While the other types of application integration are focused on the real-time exchange of information (or adherence to a common process model) between systems and companies, portal-oriented application integration is concerned with externalizing information out of a multitude of enterprise systems to a single application and interface. That's clearly an approach that goes against the notions of the other types of application integration, which are more real-time and event-driven oriented.

Application integration, while typically referring to the automated movement of information or the binding of processes between two or more applications, without the assistance of an end user, can clearly also occur at the user interface. Indeed, most examples of B2B information exchange today are also examples of portal-oriented application integration.

LINTHICUM@ATT.NET

XML-J ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
Altova	www.altova.com	978-816-1600	36
Assande	www.assande.com	212-401-2988	4
BEA	http://dev2dev.bea.com/workshop3	800-817-4BEA	2
IT Solutions Guide	www.sys-con.com	888-303-5282	17
Java Developer's Journal	www.sys-con.com/jdj	888-303-5282	23
Mindreef	www.mindreef.com	603-465-2204	9
OpenLink Software	www.openlinksw.com/virtuoso	800-495-6322	35
SYS-CON Media e-newsletters	www.sys-con.com	888-303-5282	29
The Gilbane Conference	www.lighthouseseminars.com	781-821-6734	13
Web Services Journal	www.wsj2.com	888-303-5282	12
XML Resource CD	www.sys-con.com/freecd	888-303-5282	19

WRITTEN BY **MICHAEL J. POLCYN**

Leveraging XML Knowledge to Design, Develop, and Deploy Speech Applications

Packaged apps ease the process

The voice solutions market is fueled by the rapidly growing number of consumers who want easy anytime, anywhere access to information and services via any device – wireline or wireless. With more than 1.5 billion phones and over 800 million mobile device users and 1 billion landline accounts worldwide, the telephone has become the ultimate access device to the Internet.

Enterprises are looking for new ways to reach the consumer as well as extend the information to their employees. This combination is creating a powerful force that is driving the voice business market. Additionally, vast improvements in advanced speech recognition technologies have brought new applications into mainstream use as enterprises seek to increase revenues while reducing costs. In fact, the Kelsey Group has estimated that the U.S. market for voice-activated applications and infrastructure targeting both the enterprise and carrier markets will hit $2.2B by 2006, with open standards–based solutions overtaking proprietary voice response platforms by the end of 2003 .

Our Development Challenge

As enterprises face increasing cost pressures and the need to generate additional revenues through product and service differentiation, new solutions are needed to deliver consistently high-quality customer service while improving operational efficiencies. Customer service through recent investments in Web technology tackle part of the business challenge, but this approach doesn't help customers who do not have Internet access or for whom Internet access is inconvenient or impractical.

Interactive Voice Response (IVR) products have been available for many years to automate call processing to alleviate some of the efficiency pressures affecting businesses. But until recently, these products were based on touch-tone or Dual Tone Multi-Frequency (DTMF) technology. Users often find touch-tone technology cold and unfriendly, and in general it leads to long and confusing menu structures that are not at all intuitive. Moreover, these products typically operate in a proprietary environment that requires the developer to tie learning to a specific voice response platform that doesn't leverage an understanding of Web technology. Now there are solutions that deliver the benefits of speech recognition and text-to-speech within a VoiceXML environment and can increase flexibility and simplify deployment and integration tasks.

Design, Develop, and Deploy Speech Applications

As developers we need easy-to-use and open standards–based tools that allow us to quickly and easily build advanced VoiceXML-based voice user interfaces through the dynamic generation of reusable code. These development tools do exist, and they provide the foundation for a powerful, market-proven application development environment for the creation of voice portals with speech recognition and text-to-speech applications.

The capabilities of today's VXML-based development environments now incorporate built-in, icon-based tools – from code editors and graphical dialog designers to grammar specifications and rehearsal and emulation tools – that measurably reduce the time and complexity of development, coding, and integration for customers' voice solutions. These capabilities allow you to concentrate on application call flows and the user experience, rather than on cumbersome code syntax and structure. However, when selecting a voice automation tool to begin work, care should be taken to ensure that the tool is fully compliant with the latest VXML specification approved by the WC3 and that you can capitalize on the benefits of packaged application components to speed up deployment.

The Speech-Enabled Application Developer Challenge

One of the biggest challenges I've faced in developing speech-enabled applications is in creating reusable components with enough flexibility in the Voice User Interface (VUI) to be truly reusable. All too often I've rewritten code that performs functionality that has already been written, but has to satisfy different VUI requirements. What I've learned is that though there are consistent reprompting and error-handling strategies, there is no universal VUI. Often each company, and even each department within a company, has very different VUI requirements based on the end users of the application. For example, a Personal Identification Number (PIN) reset application has basically the same functionality, but depending upon whom it is intended for, there are differing security considerations and user experience level considerations that drive what information is required from the end user and how the prompting is structured to verify identity. In reality the information gathering for user verification is a VUI requirement, but the PIN reset functionality is basically the same in

Author Bio

As a 17-year veteran of Dallas-based Intervoice, Michael J. Polcyn oversees development of new products in his capacity as senior vice president of Research and Development. Michael was a member of the team that designed and developed the first digital signal processor-based interactive voice-response product and holds several patents. Prior to joining Intervoice, Michael designed and developed a PBX-based, packet-switched LAN at Intecom; his career also includes positions with Texas Instruments and International Power Machines.

all situations. The is the niche that packaged applications can fill by providing fully configurable packages, both in VUI and data access, that can either be used as standalone applications or be incorporated easily into larger applications.

Packaged Application Components Speed Deployment

To deliver more active and effective customer service, while at the same time leveraging your knowledge of XML data, I recommend using new-generation packaged voice application components. Why? Packaged voice application components provide real-world feature functionality that literally transforms pervasive service and product delivery processes into conversations.

While some voice solutions are based on the strategy of creating vertical applications for specific market segments, you can now also leverage "horizontal" packaged application components that address the common functions needed in many industries. As you know, password resets, for example, are much the same in any market application. But these reusable modules, deployed within larger applications or with customized feature sets, will allow you to leverage the efficiencies of packaged components while delivering the unique solutions your market requires. In fact, you can deploy packaged application modules to manage crucial speech-related functions such as:

- ***Communications:*** By accessing easy-to-use prepackaged applications, you can now give callers fast, simple access to other people, places, and information resources from any telephone device at any time. These application modules offer today's most advanced features, including speech recognition support, voice activated dialing, PIN functionality, and standard interfaces to the telephony network.
- ***Location:*** Packaged applications also can be used to deliver speech-enabled locational services by phone number, street address, or postal code, country/state/zone/prefecture, or cross streets. You can customize locational applications to meet virtually any end user requirement.
- ***Identification:*** You can now use packaged modules that allow your end users to quickly and easily reset PIN and Password settings themselves. These convenient self-serve features include a consistent VUI, caller authentication, and back-end functionalities that can be customized for a wide range of industries.
- ***Surveys:*** You can also incorporate full-featured customer survey solutions within voice-driven solutions. These packaged applications will give your users the ability to create, launch, and analyze surveys, and employ a range of advanced capabilities such as rules-based activation, multiple DNIS- or identifier-based survey options, and text-to-speech or prerecorded prompt interfaces.

A Key to Mainstream Adoption

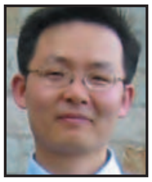
I believe that packaged voice applications offer proven benefits to the development process and to end users utilizing voice-driven solutions…and they may be a key to the widespread adoption of speech-enabled technologies.

This new generation of ready-to-use applications dramatically reduces the complexity associated with the development process, while also reducing the time, cost, and risk of designing and marketing voice-based solutions. The best of these packaged toolsets employ a process-based approach that addresses every phase of the customer service life cycle, including customer care, information delivery, transaction processing, productivity, and communications. They provide the convenience and economy of a customizable, out-of-the-box VUI, and can be used as an application template for developer solutions, and as a configurable application. Packaged modules can be used to develop applications for large enterprises, service providers, and small-to-medium sized businesses. Today's most advanced packaged applications employ data object modeling and an advanced application control environment to protect existing infrastructure investments and to streamline the integration of back-end systems. By accessing the applications' modular construction, you can easily add or remove functionality without reworking the entire solution. A good packaged application also will incorporate the business intelligence and reporting systems needed to track and analyze caller usage and behavior and overall system performance.

When packaged speech applications are deployed as a flexible, reusable component, these solutions can benefit you as a developer and the organizations you support.

MIKE.POLCYN@INTERVOICE.COM

XML for Client-Side Computing



WRITTEN BY
COACH WEI

XML is a simple, flexible text format initially designed for large-scale electronic publishing. It is flexible, open, and human-readable, and can be learned easily. XML can also be generated, parsed, analyzed, and transformed easily. It's no wonder that XML has been widely used for server-side computing: J2EE, .NET, and Web services.

However, we have not seen significant use of XML on the client side to date. When we write client-side code, we are likely using HTML/DHTML for browser-based applications, Win32 for Windows desktop applications, and Java Swing/J2ME for Java applications.

The truth of the matter is that XML makes a lot of sense for client-side computing. The difference between client-side computing and server-side computing is that the former is more concerned with user presentation and interaction while the latter is more concerned with business logic and data access. XML has proven to be an effective tool at both the business logic layer and data layer. It is also a great solution to the challenges at the user interface (UI) layer. Two UI layer challenges stand to benefit most obviously from the use of XML.

The first challenge is how to decouple UI description and UI logic. Separating UI description and interaction logic would allow greater flexibility, clearer separation of programmers and designers, and lower development and maintenance costs.

Today, UI description and UI logic are tightly coupled in most client-side programming models. UI description must be written as program code and normally mixed together with UI interaction logic. For example, to create a UI layout for a Win32 dialog, a developer must write Win32 code. Changes to the UI layout would require changing the code, recompiling the program, and redeploying the program. The cost and complexity of this process are high.

XML is an ideal candidate to solve this challenge. UI description can be written as XML documents and UI logic can be written in any programming language. Such separation decouples them and creates tremendous flexibility. XML documents can be created and manipulated by many tools and are human-readable. Additionally, designers with HTML skills will have no problem dealing with XML documents. UI designers can create and edit such documents without touching the UI logic. UI developers are free to code UI logic without getting bogged down in UI design. Such clear separation would significantly lower development and maintenance costs.

Additionally, using XML for UI description would enable better development tools. The best development tools enable visual design of user interfaces in a drag-and-drop fashion. However, these tools are merely "UI code generators" for developers and preclude the participation of people other than developers. For example, Visual Studio .NET generates UI description as C# code and only C# developers have the skill sets to use such code. Furthermore, when developers modify such C# UI code, it is a daunting task for the development tool to parse the C# code, figure out what has been changed, and determine how the change should be reflected in the visual design. With XML for UI description, it would be easier to create tools that enable two-way editing of UI design for nonprogrammers and allow programmers to concentrate on business logic.

The second challenge is how to decouple the UI layer from the underlying platform or device. Companies will always have different computing platforms. If applications can run on any browser, operating system, and hardware device, the cost for development, deployment, maintenance, and support will be dramatically lower.

It's easy to see how business logic can be made platform independent, but it is much harder for UI. Even Java did not adequately solve this challenge. As a result, most client-side programming models are tied to the client-side platform, such as Win32, MFC, Java Swing, and DHTML.

XML is inherently cross-platform. UI designers can use XML to describe their design and the cross-platform nature of XML will enable such information to be communicated accurately to different platforms. Different platforms can choose to present such UIs in a platform-optimized way without requiring UIs to be hard coded into this platform. UI logic can be easily compiled or coded to run cross-platform using available technologies today, such as Java or JavaScript. Even if there are incompatibility issues, they can be resolved by leveraging the flexibility of XML (e.g., XSLT).

Client-side XML is beginning to draw attention. There have been efforts and successes from standards bodies, and startups are beginning to deliver real commercial implementations. Scalable Vector Graphics (SVG), an XML standard for 2D graphics over the Web, is a W3C standard and has gained a lot of support in the industry. XUL, an XML language for describing rich user interfaces, developed by Mozilla, significantly increased awareness of client-side XML. On the commercial implementation side, there have been quite a few successful examples. For example, Starwood Hotels developed a customer response system across its hotel chain (www.nwfusion.com/ee/2003/eecrm.html). The user interface of the application is written using XML and the UI logic is written as client-side beans. It runs inside any 4.0+ browser on major platforms and deploys like a normal Web application with zero client install, but delivers native desktop application performance and functionality. The application has been running nonstop for 14 months, supporting thousands of users distributed across the country. The results were significant: shorter development cycle, huge deployment and maintenance savings, and higher user satisfaction and productivity. It was cited as one of *InfoWorld*'s "Top 100 Innovations" in 2002.

What started as a great success on the server side is just now beginning to show its strengths in client-side programming. We have yet to see the full potential of client-side XML, but can expect to see much more as it drives a client-side revolution.

CWEI@NEXAWEB.COM

AUTHOR BIO

Coach Wei currently serves as CTO for Nexaweb (www.nexaweb.com), which develops the leading software platform for building and deploying Enterprise Internet Applications. Previously, he played a key role at EMC Corporation in the development of a new generation of storage network management software. Coach has his master's degree from MIT, holds several patents, is the author of several technology publications, and is an industry advocate for the proliferation of open standards.

Introducing OpenLink Virtuoso

A **CROSS-PLATFORM Universal Server** that transparently integrates disparate application databases and disparate applications across your enterprise. It provides a Virtual Database, XML Database, SQL Database, Web Application Server, and Web Services Platform as part of a **single-server solution.**

Empower your organization today, download a FREE evaluation copy from www.openlinksw.com/virtuoso or call us on 1 800 495 6322

Database Support
Oracle, SQL Server, DB2, Sybase, Informix, CA-Ingres, Progress, MySQL, PostgreSQL, and other ODBC or JDBC accessible databases.

Protocol Support
SQL, ODBC, JDBC, XML, XSL-T, XQuery, XPath, HTTP, WebDAV, NNTP and POP3

Platform Support
Windows, UNIX (all major flavors), Linux, Free BSD, and Mac OS X

Programming Languages
Any .NET bound language, Java, C/C++ etc.

Web Application Hosting
ASP.NET, Java Server Pages, PHP, Perl, and Python

Runtime Environment Hosting
Microsoft .NET, Mono, Java, Perl, Python, PHP, and others

www.openlinksw.com/virtuoso

OPENLINK® SOFTWARE

www.altova.com

xmlspy® 2004

NEW Microsoft® Visual Studio®.NET Integration
Enhanced Database Connectivity
XML Differencing
XPath 2.0 Analyzer

improved XML Schema/WSDL Editor
XSLT Debugger and Code Generator

Introducing mapforce™ 2004

New Visual Database-to-XML and XML-to-XML mapper!

Generate Java, C#, C++, and XSLT

mapforce™ 2004 FREE 30 day Trial pass

*Accelerate your next XML development project with the leading XML tools from Altova! **xmlspy® 2004** is the newest release of the industry standard XML Development Environment for modeling, editing and debugging any XML technology. **mapforce™ 2004** is an all-new, visual data mapping and code generation tool. Specially priced product bundles are available, see website for details.*

Download xmlspy® 2004 and mapforce™ 2004 today: www.altova.com

ALTOVA®
www.altova.com