Enterprise Solutions • Data Management • XML Labs

XML JOURNAL

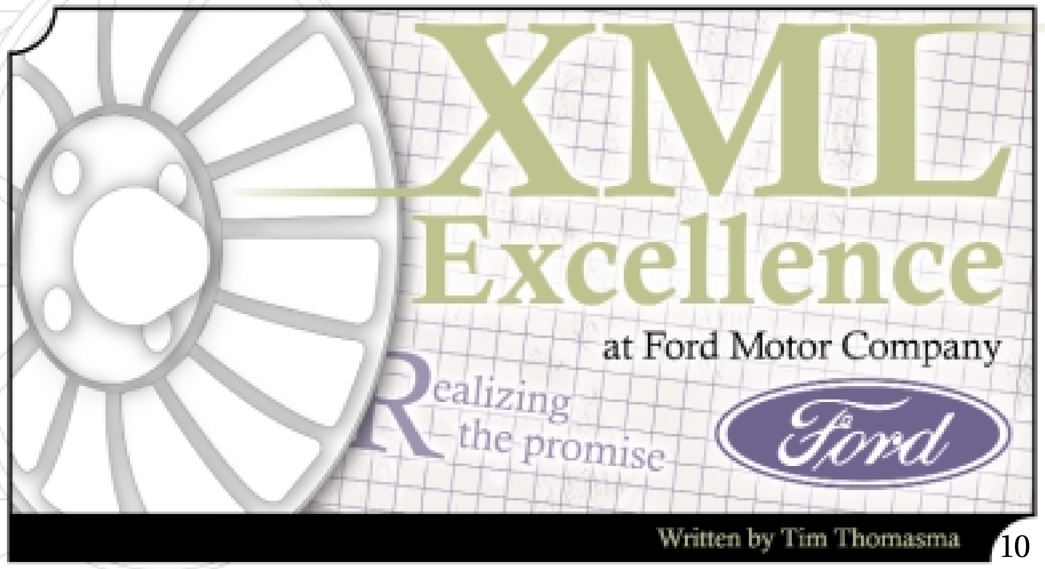
THE ULTIMATE XML ENTERPRISE RESOURCE

September 2002 | Volume:3 Issue:9

XML-JOURNAL.COM

WEB SERVICES EDGE 2002
FREE
EXPO PASS!
$75 VALUE
OCTOBER 1-3, 2002
SAN JOSE McENERY CONVENTION CENTER, SAN JOSE, CA
INSIDE PAGE 33

SYS-CON MEDIA



BEA

dev2dev.bea.com/useworkshop

XML JOURNAL

FOUNDING EDITOR
Ajit Sagar ajit@sys-con.com

EDITORIAL ADVISORY BOARD
Graham Glass graham@themindelectric.com
Coco Jaenicke cjaenicke@attbi.com
Sean McGrath sean.mcgrath@propylon.com
Simeon Simeonov sim@polarisventures.com

EDITORIAL
Coeditor-in-Chief
John Evdemon jevdemon@sys-con.com
Coeditor-in-Chief
JP Morgenthal jpm@sys-con.com
Editorial Director
Jeremy Geelan jeremy@sys-con.com
Executive Editor
M'lou Pinkham mpinkham@sys-con.com
Managing Editor
Cheryl Van Sise cheryl@sys-con.com
Editor
Nancy Valentine nancy@sys-con.com
Associate Editors
Jamie Matusow jamie@sys-con.com
Gail Schultz gail@sys-con.com
Jean Cassidy jean@sys-con.com
Assistant Editor
Jennifer Stilley jennifer@sys-con.com

PRODUCTION
VP, Production & Design
Jim Morgan jim@sys-con.com
Lead Designer
Aarathi Venkataraman aarathi@sys-con.com
Art Director
Alex Botero alex@sys-con.com
Associate Art Directors
Cathryn Burak cathyb@sys-con.com
Louis F. Cuffari louis@sys-con.com
Richard Silverberg richards@sys-con.com
Assistant Art Director
Tami Beatty tami@sys-con.com

CONTRIBUTORS TO THIS ISSUE
Len Bullard, Mark Feblowitz, Tom Gaven, Sig Handelman, Kenneth J. Hughes, Ian Moraes, Uche Ogbuji, Michael Rowell, Suresh Selvaraj, Hitesh Seth, Karl B. Schwamb, Pat Snack, Hans Tallis, Tim Thomasma, John Walker

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.

Stick It in Your Ear

WRITTEN BY **JOHN EVDEMON**

As the 1970s drew to a close, Douglas Adams's "The Hitchhiker's Guide to the Galaxy" first appeared on BBC radio. "Hitchhiker's Guide" was (and continues to be) wildly successful – the series was adapted into a four-book "trilogy" and a hit TV show. "The Hitchhiker's Guide to the Galaxy" featured a wonderful mix of Monty Python–like humor, science fiction, and science fact. One of the most intriguing ideas offered was the now infamous Babel fish. According to the book, the Babel fish "absorbs unconscious mental frequencies from the brainwave energy of those around it and excretes into the mind of its carrier a telepathic matrix formed by combining the conscious thought frequencies with nerve signals picked up from the speech centers of the brain that has supplied them." In simpler terms, putting a Babel fish in your ear enables you to instantly understand anything said to you in any language.

Surprisingly, some people have claimed that XML can provide similar benefits (fish insertion optional). "Communicating data in an XML format enables data to be freely shared or exchanged in an open manner." How often have we heard that claim? Sadly, XML-based integration isn't as easy as the marketing folks may have led us to believe. Numerous published articles have explained how XML can be used to extend or replace e-business infrastructures. While it has gained significant ground within the enterprise, XML has yet to match the success of EDI as the syntax of choice for e-business. The vast majority of all B2B transactions continues to use EDI. Critics claim that EDI uses a cryptic, difficult-to-understand syntax and is expensive to implement. Given these claims, the costs associated with EDI may seem to outweigh the benefits. How can EDI be so successful if it's so expensive and difficult to use? There are two answers to this question – one simple, one not so simple:

1. *Most of the EDI critics are wrong.*
2. *EDI defines standardized semantics for specific verticals.*

While the first answer may surprise (or infuriate) many people, I'd like to concentrate on the second (e-mail me if you'd like to discuss/debate answer number one).

Two "versions" of EDI are available: ANSI X12 and EDIFACT. Both are managed by large standards organizations – ANSI and UN/CEFACT – that have divided their organizations into several domain-specific working groups. Each working group is chaired and populated by domain-specific experts who define and maintain the messaging standards used by X12 and EDIFACT, which are then published and distributed throughout the world. EDI (X12 and EDIFACT), then, consists of a clearly defined set of syntax and semantics for specific domains.

XML has a clearly defined syntax but lacks the sort of semantic transparency made possible by EDI's X12 and EDIFACT metadata. For example, one company may define an invoice as INVOICE while another uses INVOICE_810 (based on an X12 representation), and a third uses BILL. To a human observer (especially one familiar with X12) these elements all refer to the same concept, an Invoice. A machine, however, has no way of knowing that INVOICE, INVOICE_810, and BILL all represent an Invoice.

So how can this issue be avoided? One possible approach requires companies to exchange their schemas in advance of conducting business (this approach is similar to how EDI Implementation Guides have been used for decades). Another approach is to adopt an existing XML initiative and extend it to suit your company's requirements (again, much like EDI). Many of these XML initiatives are continuing to evolve – XCBL has been adopted by OASIS's UBL (Universal Business Language) initiative while OAG recently released version 8.0 of OAGIS (see Mark Feblowitz's excellent article in this issue). Both UBL and OAGIS utilize component-driven approaches, enabling users to define, assemble and extend schemas as needed. Companies can also simply define their own schemas using their internal sets of metadata (this may, in fact, be the best option for many firms).

A final option leverages a riskier, yet compelling option – the use of emerging Semantic Web initiatives for semantic integration. This approach might utilize RDF and RDFS to represent the metadata used to characterize and establish relationships between system components. This last approach appears to be the future for both XML and semantic integration challenges.

Tim Berners-Lee's vision of the Semantic Web is rapidly maturing – the convergence of grid computing, agent-based architectures, and Semantic Web technologies is helping to make his original vision a reality.

ABOUT THE EDITOR

John Evdemon is coeditor-in-chief of XML-Journal *and CTO of DiscoveryLogic (www.discoverylogic.com). He is also an Invited Expert with the W3C XML Core Working Group.*

JEVDEMON@SYS-CON.COM

ADOSCO.,LTD.

http://www.a-dos.com

SYS-CON MEDIA

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

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

BUSINESS DEVELOPMENT
VP, Business Development
Grisha Davida grisha@sys-con.com

COO/CFO
Mark Harabedian mark@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
Advertising Account Manager
Megan Ring-Mussa megan@sys-con.com
Associate Sales Managers
Carrie Gebert carrieg@sys-con.com
Kristin Kuhnle kristin@sys-con.com
Alisa Catalano alisa@sys-con.com
Leah Hittman leah@sys-con.com

SYS-CON EVENTS
VP, Events
Cathy Walters cathyw@sys-con.com
Conference Manager
Michael Lynch mike@sys-con.com
Regional Sales Managers, Exhibits
Michael Pesick michael@sys-con.com
Richard Anderson richarda@sys-con.com

CUSTOMER RELATIONS
Manager, Customer Relations
Anthony D. Spitzer tony@sys-con.com
Customer Service Representative
Margie Downs margie@sys-con.com

JDJ STORE
Manager
Rachel McGouran rachel@sys-con.com

WEB SERVICES
Webmaster
Robert Diamond robert@sys-con.com
Web Designers
Stephen Kilmurray stephen@sys-con.com
Christopher Croce chris@sys-con.com
Catalin Stancescco catalin@sys-con.com
Online Editor
Lin Goetz lin@sys-con.com

ACCOUNTING
Assistant Controller
Judith Calnan judith@sys-con.com
Accounts Receivable
Kerri Von Achen kerri@sys-con.com
Accounts Payable
Joan LaRose joan@sys-con.com
Accounting Clerk
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: $77.99/yr (12 issues)
Canada/Mexico: $99.99/yr
all other countries $129.99/yr
(U.S. Banks or Money Orders)
Back issues: $12 U.S. $15 all others

Battle of the Bulging Standards

WRITTEN BY **UCHE OGBUJI**

The many people and organizations who came to the XML industry from the database and software development industries have always wanted better standards for modeling the native data structures they interchange in XML. Some support for this was always likely in developing the XML Schema language that was expected to supplant XML 1.0 DTDs. Yet when drafts of the W3C XML Schema language (WXS) emerged, the mechanism provided for integrating with data type support proved immediately controversial.

To simplify the matter a bit, the Post Schema Validation Infoset (PSVI) annotates the Infoset of an XML document (i.e., the abstract model of its nodes) with information about the types that became associated with each node during schema processing. The many people who came to XML from the document and text management space, and even from the Web architecture space, objected to this imposition of programming types into XML at such a fundamental level. The main fear was that this would end up causing a lot of complexity in other XML technologies even if people didn't use XML as a mere data serialization format.

The introduction of the PSVI is one of the reasons that so much interest persisted in alternate schema languages. RELAX and TREX, which merged to form RELAX NG, limited themselves to validating documents. RELAX NG allows one to assign data types from plug-in libraries to simple element content and attribute values, but it doesn't provide a model of any sort for these types. Such a model would have to be layered upon the process that does the validation. This difference has led to a palpable split in the XML community between those who can't imagine XML without built-in provisions for data typing and those who can't imagine why such things shouldn't be layered very distinctly from XML's textual core. Furthermore, one camp sees data typing as needed for XML tools, while the other worries that the added complexity will make XML impossible to process without sophisticated (and thus expensive) tools. The open source versus commercial software battle thus enters into the mix.

There is actually more to the choice between RELAX NG and WXS than data typing: RELAX NG has gained unexpected momentum largely because it's a much earlier tool for modeling common XML constructs. However, the debate between XML-as-pure-text and XML-as-serialized-data has become more prominent as the PSVI wends itself into other specifications. The emerging new battlefield is the W3C XML Query language and its effect on the next iterations of XPath and XSLT.

XPath and XSLT are arguably the most successful XML-related technologies besides XML 1.0 itself – and they're relatively simple, providing ways of addressing the nodes that make up XML documents pretty much as they are after basic parsing. Some facilities, such as the sum() function in XPath and <xsl:sort/> in XSLT, do provide for superimposing operations based on data types, but these are lightweight. Users who take to WXS and thus end up using its sophisticated data-typing system, including such things as dates and floating-point numbers, wish for more tools for manipulating these in XPath and XSLT.

The XML Query working group early on recorded a requirement for support of WXS types. XQuery modeled itself as a sort of supercharged extension of XPath node access and XSLT result templates, and eventually the W3C decided to pool the effort toward XQuery 1.0, XPath 2.0, and XSLT 2.0. The result of all this was that next-generation XPath and XSLT drafts became much more complex as they included all manner of facilities for PSVI and data types.

People in the XML-as-text camp, who at first simply wrote off XQuery as a byzantine curiosity that they needn't worry about, suddenly found their beloved XPath and XSLT transformed into exactly the kinds of specifications they feared the PSVI would inspire.

The result of all this right now is a pitched battle between the text and data types camps, with XQuery held up as the model for all that is supposedly good or bad about the directions in which XML is moving. XQuery itself has been undergoing changes to try to accommodate all the competing concerns. And comments on most of XML's public mailing lists, including that of the W3C technical architecture group, intimate that a showdown may be in the offing as the drafts in question progress toward recommendation.

XML technology has brought together an impressive diversity of practitioners. Now a debate between the two prominent branches is destined to shape its future.

AUTHOR BIO

Uche Ogbuji is cofounder and CEO of Fourthought, Inc., a software vendor and consultancy specializing in XML solutions. A computer engineer, he has worked with XML for several years, codeveloping 4Suite, an open-source platform for XML processing. A frequent conference speaker, Uche has authored a number of articles on the practical use of XML.

UCHE.OGBUJI@FOURTHOUGHT.COM

SONIC SOFTWARE

www.sonicsoftware.com/websj

Reader Feedback

Slightly Swayed...

Tom Gaven makes some sound points ["Desperately Seeking...Help for XML Schema," June 2002].

Yes, XSD is a complex beast and much needs to be fixed in Version 1.0....Can we not get behind this great effort to help improve it? Because even he acknowledges that there is much good and useful in XML Schema.

The brevity of a non-XML syntax for a schema language is alluring but much is lost as well, the biggest loss being the ability to use an XML-aware tool on the schema itself, to transform it, to render it, to intelligently document it. I'm not yet convinced that a regression to a non-XML syntax is a good thing.

At present I'm content with using a simplified subset of XSD as a base and augmenting its limitations with alternate technologies such as XSLT, Schematron, etc. *And letting the "smart" folks at the XSD Working Group know about it.*

Nevertheless, I enjoyed the article and hope to read more from Tom in the future.

NAUMAN MALIK
via e-mail

The Author Replies

I acknowledge that XML Schema is version 1.0. Of course, it was in development for a couple of years (probably a mistake) and had many, many big companies in on the development (probably a bigger mistake). I certainly think the problems can be fixed, but I hope this means removing features, not adding in even more complexity for the next version.

Also, I wouldn't necessarily consider a non-XML syntax (e.g., RELAX NG's compact syntax) a "regression." As long as there are conversion tools available to go from the compact syntax to/from the XML syntax, we get the best of both worlds. The XML syntax enables all the benefits of XML tools, editors and processing. The compact syntax makes it much easier for humans to author and understand schemas. Of course, the conversion to/from the two syntaxes must be "lossless" for this to be completely effective.

Letters may be edited for grammar and clarity as well as length. Please e-mail any comments to John Evedemon (jevdemon@sys-con.com) or JP Morgenthal (jpm@sys-con.com).

Walking on Bones

WRITTEN BY **LEN BULLARD**

Consider that when the pharaohs were buried with the wealth of Egypt, tomb raiders did the economy a favor by liberating that wealth and putting it back into circulation. But at the end of the day they were grave robbers who walked on bones not to achieve any greater good than enriching themselves. We should ask ourselves if some parts of the Web experience have been examples of tomb robbing.

While many do serve tirelessly and well, the vendor-centric nature of the consortia self-selected to create specifications for global systems inclines to self-enrich. Indeed, that is their corporate responsibility, just as it was the responsibility of the priests to enshrine the pharaoh's jewel-encrusted corpse with the wealth of the nation, conveniently declared as the pharaoh's wealth for the afterlife. In the religious theory of the time, the dead pharaoh would serve as intercessor with the gods for mankind's benefit, and the wealth would sustain him in that role. In practice, a lot of gold ended up gathering dust next to a sarcophagus with no provable benefits to anyone this side of the embalmer's nose pick.

Accept for argument's sake that royal tomb building had a purpose. Through the authority of a pharaoh, society took direction and could be organized to create massive public works. The problem was, the task chosen by the authority organized the society but didn't provide a coherent means for sustaining the process except insofar as the next pharaoh copied processes of the last one and workmen trained apprentices.

Meanwhile, gold was being left in the tombs for robbers. Burying the next pharaoh became a problem for the priests. In the case of Tutankhamen, priests raided the tombs of his father and mother. Because Akhenaton and Nefertiti's legacy was to be obliterated, given their monotheistic heresies, this presented an opportunity to continue the process of tomb building and to reestablish the priesthood that had been defrocked during Akhenaton's reign. Walking on bones suited their purposes. It wasn't an innovation. Akhenaton had desecrated the monuments of Egypt, including that of his father, to get rid of the names of other gods. Tutankhamen, before being dispatched himself, had returned Egypt to its former cultural beliefs and the priesthood back to primacy. These actions didn't further the economy of the nation, but they did serve as an object lesson to the priests: pharaohs aren't always considerate and, living god or not, had to be controlled. Both the gold and what was said in the hieroglyphs were sources of the power of the priests. They had to prevent another Akhenaton at all costs.

So what?

Even before the dot.bomb went off in the faces of the investing community, some had become uncomfortable with the myths that were promoted about the origins of the Web and the technologies it depends on. In the rush of Internet Time, the origins of the Internet technology in the research work of the U.S. Department of Defense, the advanced work in markup-based hypermedia as part of the global CALS defense initiatives, and the pioneering work of those such as Douglas Engelbart were pushed aside as irrelevant precursors of the inventors of the Web technologies. In effect, these inventors had done as so many before them had: built on the works of their predecessors, extending a bit, removing bits, and, as in the case of HTML, abandoning progress made in favor of an easier to implement, faster to field, but less powerful early version of that work. Just as the tomb raiders and priests of old Egypt had done, when it was time to get the gold, walking on bones was an acceptable practice.

So while the press rushes to crown authors such as Lawrence Lessig for his insights into the way in which powerful interests are corrupting Web technologies – indeed, robbing the so-called commons of its birthright – they carefully avoid peering into the historical trends that accompanied this fielding, ignoring the early work done by the well-funded institutions of government and industry that provided the ground for others to build over. They are the classic example of those who cannot remember history because it doesn't work for their own agenda, and are themselves creating the very corruption they rant against.

It's a problem of choosing and consenting to authority. The protocols that make the Internet a reality are themselves cultural agreements on what is to be called by what name, how to behave, given such names, and what is to happen if a violation occurs. While a protocol can clearly be stated and violations detected, what to do when a violation occurs is a matter of authoritative prescription. The principle of equality of all before the law is fundamental to this notion. We simply can't replace one monopoly with another regardless of the public perceptions of these entities. This is the mistake that Lessig, the open source community, and those who would give up their own choices to the consortia make. The dominance of the Internet Explorer browser is seen by some as a

Continued from page 7

cunning plot by Microsoft to rule the Web. The same people usually fail to note that IE is itself stunningly innovative and reliable. They decry the entry of commercial interests into the Web, but fail to notice that content and capability as well as penetration of the technology into homes have increased as business interests spent their resources to promote the Web and market the technology, and continue to give away the browser technology.

This doesn't come without a price. IE is tied to the Windows operating system. A virtual lock-in has occurred. Is the price of a reliable Web admitting that in a world of relentless innovation, some piece must be stable, and to keep it stable, having it under the control of the company that builds it is currently the way this is done? Is the answer the same for the browser as it is for the Java language? While populists attack the record companies in the name of fair use, does the uncontrolled copying of copyright works onto a medium that allows unlimited access constitute true fair use? Who gets to say?

If we ground our meanings in authorities, what do we do when authorities use these meanings to grind us into dust to ensure that their meanings and their corpses are enshrined? What forces compel us to walk on bones? Failing to understand and limit authority – and forgetting the past – limits our ability to sustain. If we want other than consortia and vendor lock-in, we must invoke the rule of law as the highest authority. Given the perception that the U.S. government has capitulated to Microsoft, some believe the future is pessimistically preordained to favor big interests. Others may note that Microsoft hasn't escaped without penalty, that the case proved among other things that others' claims to the rights of hegemony over the Web are weak, if not weaker, and that, in effect, the game that Microsoft played was the same as its competitors – including the open source community – were playing, but that Microsoft played better.

Spilt milk. None of these factions has the *right* to govern the future of the Internet. It is, as Lessig claims, a commons, but the right to govern is of the people, and for this a representative form must be enabled. One of the worst mistakes of the Web generation of developers was to empower the W3C with unwavering support, thus privatizing these interests and concentrating them into an organization of vendors. What Microsoft, Sun, IBM, Oracle, and their like could not have accomplished by any other means, the Web generation handed them as a free gift. In short, Web developers have been naive and the press unscrupulous, and commercial interests have done with the first two groups precisely what one would expect: recognized a good thing and run with it.

What can be done? Not a lot quickly. Experience shows that changes will come from establishing standards based on proven practice and not specifications for untested systems. Legitimate authority comes from legitimate sources whose polities are provenanced by our representative forms of government. If we are to have a stable, innovative, and progressive community, we would do well to acknowledge that our knowledge of history must be as complete as our mastery of the algorithms of computer science else the claims of invention cannot be met with proofs of prior art. We have to be the shepherds of development, but also sheepdogs, vigilant, capable of nipping at the heels of the herds. We must acknowledge that the companies that are trying to entomb themselves with the wealth of the Internet are only doing as others (including ourselves) do, and that we're tied together in this. Ultimately, teams succeed and overcome competitors. In choosing an authority, we choose a team. It's time we looked beyond our companies and our consortia and realize that our future does depend on how we apply the principles we espouse.

Here are some thoughts on potentially self-defeating trends in our environment:

- **Be wary** of the expanding roles of private vendor consortia in creating specifications for public services. Regardless of the reputations of the public leaders, over time, tremendous power over global information assets is concentrating in the hands of a few corporations with little or no oversight.
- **Be wary** of the "programmers first" mentality that has pervaded the Web since its inception. XML is a middle ground where the programmer and the subject matter expert meet to describe information. XML has been on a reckless path of development of increasingly complex and obscure specifications that can easily undo the previous decades of progress in the markup industry toward freeing information from proprietary vertical applications.
- **Be wary** of the "kill all the lawyers" mentality. The naivete and inexperience of the programming community with regard to the political and social impacts of their work further the complexity, the ownership by the consortia, and the increased power of a few vendors. Despite laudable goals, the Web community's early deprecation of global standards organizations such as ISO made this possible. Ends do not justify means. Seizing and privatizing markup technologies has an inevitable and sad effect: the pharaoh owns the gold, and if he chooses to be buried with it, we'll have to walk on his bones to get it back.
- **Be wary** of the press. This industry has a long history of forgetting history. They make money by reporting news, and last week's consensus is too boring to be newsworthy.
- **Be wary** of trusting the Web. One can learn by experience to trust people, but a verification system based on checking opinions is a gossip chain. It's too easy to use propaganda to create a consensus of the uninformed.
- **Be wary** of those who say, "If it doesn't have a URI, it isn't important." Don't ignore the local library. The existence of URIs or the lack of them in a card catalog doesn't erase the value of the past. The Web is littered with self-proclaimed inventors who merely copy without attribution.
- **Be wary** of the priests. The network effect and critical mass are not intelligent means to make local choices. Opinion leaders have agendas too. Think for yourself, not to contend, but to understand. The priests may be right and it may be time to rob a tomb, but they may just be too lazy to haul gold from the mines.
- **Be wary** of XML namespaces. While it's advisable to share definitions, an XML namespace with a URI includes a notion of authority and implicit or explicit ownership. An authority can be decisive without owning the gold.

XML is the middle ground. The reliance on syntax over semantic agreements as the foundation is a starting place for negotiators. The hieroglyphs remain wonderfully consistent from tomb to tomb. Still, if we are to keep the gold in circulation, we must have a publicly supported means to mine gold and a right to get the gold and use it. The system will not self-correct out of necessity but because careful attention is paid to correcting it.

We must realize that when Tim Berners-Lee said the Web would be a reflection of society, we became interdependently responsible for that image in the mirror. The mirror can't care and still do its job. The Web is a human-maintained projection of our expectations and desires, not an empirical source of truth. We have to be careful about distinguishing between assigning power to do tasks and authority to choose tasks to do. The Web is a tool and a resource. The W3C is a vendor consortium. Neither is the government.

It's okay to organize priests around building shrines. It's not okay to lock up their pay in the vault with the honored corpse. It's the duty of the pharaoh to organize the activities of the people. It is not the right of the pharaoh to own and dispense the profits of the labor so ordered. Equality of laws, or of people before the law, includes the pharaoh. This was the fundamental change from the divine rights of kings to governance of the people, by the people, for the people. The Web and the consortia kneel before the law.

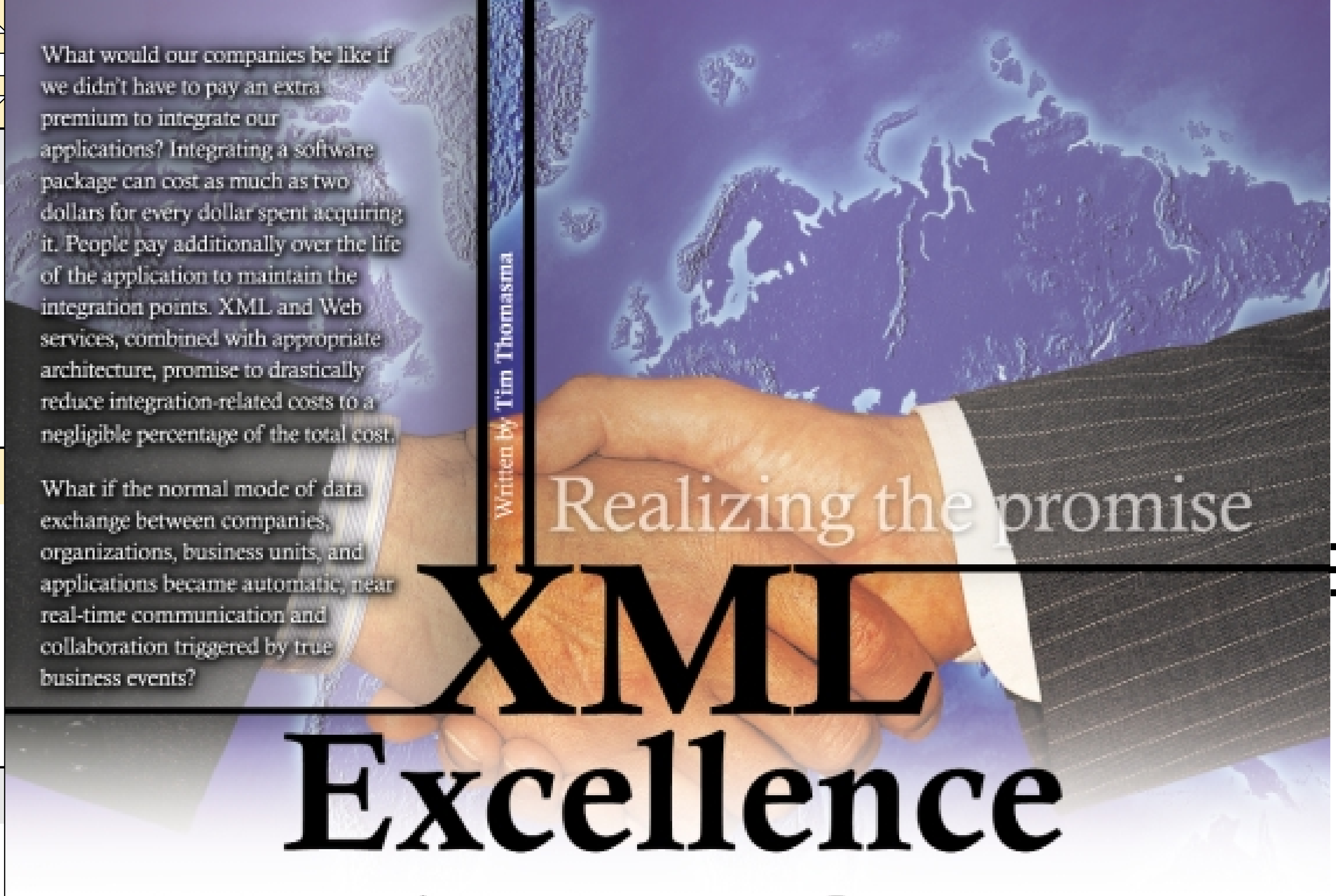
AUTHOR BIO

Len Bullard is a veteran markup specialist employed as a systems analyst by the Public Safety division of the Intergraph Corporation. A working musician, you can find his music at www.mp3.com/lenbullard.

CLBULLAR@INGR.COM

Macromedia

www.macromedia.com/goCFMXlight



What would our companies be like if we didn't have to pay an extra premium to integrate our applications? Integrating a software package can cost as much as two dollars for every dollar spent acquiring it. People pay additionally over the life of the application to maintain the integration points. XML and Web services, combined with appropriate architecture, promise to drastically reduce integration-related costs to a negligible percentage of the total cost.

What if the normal mode of data exchange between companies, organizations, business units, and applications became automatic, near real-time communication and collaboration triggered by true business events?

Written by Tim Thomasma

Realizing the promise XML Excellence at Ford Motor Company

For years business practices have been arranged around file transfer technology and overnight or periodic scheduled batch processing runs. Using this approach, information generated by business events isn't acted on immediately, but aggregated for later communication. An artificial event triggers the actual communication as a batch (e.g., the nightly batch-processing window for a particular application that starts at 2 a.m. Eastern Time). Information doesn't flow in a relevant way. Delay is built in. People work around this, using telephone, fax, and manual data entry to compensate for the timing-related deficiencies of the system. In addition, these practices create further problems in requiring a large number of data points to be processed at once, which in turn often requires planned downtime, which among other things obviates 24/7 operation.

XML and Web services promise to support new rapid-response business practices: as soon as a customer returns a product for repair, the information generated from this business event can be sent automatically to manufacturing, design, customer service, and finance people as well as to systems and databases – and those of the suppliers – so that all the appropriate resources of the extended enterprise are immediately engaged in satisfying this customer now and in the future.

People are looking for a universal integration back plane built on Internet technology that every application and computing device plugs into. This integration back plane would enable reliable, secure transmission of information in widely understood formats. Instead of handcrafted point-to-point interfaces and integration connections, we need standard connection points that are well known throughout the computing industry and supported by all products. This is XML excellence, applied to integration.

We think the promise is achievable. Most of the pieces are in place. We see several elements of XML excellence that are involved in realizing the promise:

- Use the overall structure and integration best practices of the Open Applications Group Integration Specification (OAGIS).
- Use and support relevant standards.
- Manifest excellence in a focused solution that delivers business value to the company.
- Drive use and compliance internally and externally to the company.
- Plan for broad deployment of these techniques – event-driven messaging, Web services, and richer electronic collaboration.

Standards are critical elements of the vision. Effective standards yield global, cross-industry coalescence around specifications that everyone implements. We understand the standards, and we work with the IT industry and other end users of IT like ourselves to drive relevant standards. We aggressively implement to the standards in ways that make us a more effective automotive manufacturer. This is part of Ford Motor Company's investment in IT excellence as a corporate strategy.

OAGIS

In April the Automotive Industry Action Group (AIAG) sponsored an interesting meeting of automobile manufacturers. Participants from Ford Motor Company, Volkswagen, Nissan, Toyota, Honda, General Motors, and DaimlerChrysler described in turn which XML standards their companies had implemented or planned to implement. There was a remarkable consensus on the Open Applications Group Integration Specification. Each company uses it or intends to. There was no similar consensus on any other library of XML document definitions and associated semantic and process definitions. Nothing else even came close.

The U.N. Centre for Trade Facilitation and Electronic Business (UN/CEFACT) has recommended OAGIS as a temporary document syntax for payloads using the ebXML infrastructure, while its new ebXML Core Components specification proposal goes through standardization. In addition to the UN/CEFACT, the Open Applications Group, Inc. (OAGI), maintains close relationships with the Organization for the Advancement of Structured Information Systems (OASIS) and the Web Services Interoperability Organization (WS-I). The OAG, OASIS, and WS-I consortia generally have representatives from the same software companies on their governing boards.

From the start, OAGIS was designed for use across all industries and in every geography. The software companies that built it, including JD Edwards, Oracle, PeopleSoft, QAD, and SAP, intended to use it to reduce the costs of application integration for their customers, no matter what industry or country their customers do business in.

We try to avoid creating our own application-specific or Ford Motor Company–specific XML document definitions for application integration or electronic collaboration with trading partners. We'd rather use open, freely available document definitions that are in widespread use. The more widely used, the better, across industries and geographies. This is because we're a global company participating in a global supply chain that interacts with other industries, including electronics, chemicals, and metals. By using a single specification as our starting point, we're able to realize efficiencies in time and resources needed to do integration and B2B projects. The benefits are sharply reduced when we have to evaluate or map between competing specifications of business meaning and process content.

OAGIS doesn't cover all the functional domains that businesses are concerned with. Other cross-industry sets of document definitions address functions such as human resources. In addition, there are vertical industry requirements. The information that a chemical company needs to include in its purchase order may be different enough from what an electronics company needs that a different document definition is required. OAG works closely with other organizations, whether they're industry horizontal or whether they're focused on a vertical supply chain or trading community.

As a result of working together with other groups, OAGIS includes mechanisms for creating horizontal and vertical extensions. The basic idea is laid out in Figure 1.

In this model there's no need for publishers of cross-industry specifications such as the HR-XML Consortium (www.hr-xml.org) or Transentric (www.transentric.com/products/commerce/tranxml.asp) to merge with or give their work to OAG. Similarly, there's no need for industry verticals such as AIAG or STAR (Standards and Technology for Automotive Retailing) to give their industry extensions to OAG. The organizations make joint development, ownership, and publication arrangements that suit the objectives of all participants. The result is effectively a single open set of document definitions available for use royalty free with the same architecture and documentation structure, built on the same data dictionary and set of core components, with consistent extensions.

Ford Motor Company was among the first end-user companies to join OAG. We joined the end-user members in challenging the vendors to bring to market products that use OAGIS, rather than compete by locking customers into closed, proprietary interfaces. In response, several companies were able to rapidly implement the specification to the degree that, three months after our challenge, they could demonstrate an implementation of a B2B scenario that we contributed. These companies came to Ford Motor Company's iTek building in summer 2000 to demonstrate XML-enabled integration to our employees and executives. Seven different software companies shipped servers to our facility, set up a local area network, plugged their applications together, and on the next day were ready to show people automatic, near real-time communication and collaboration triggered by true business events. This proved to be the first of several similar proof-of-concept demonstrations done that year and the next by AIAG, ebXML, OAG, and RosettaNet.

Hundreds of software companies offer OAGIS implementations with their products. It's becoming common to find OAGIS bundled in each software package's XML or Web services module at no additional cost.

Essential Standards

The W3C's XML 1.0, XML Schema Definition (XSD), and XML StyleSheet Language (XML/XSLT 1.0) recommendations provide the basic syntax that other specifications use. Parsers and other software implementing these specifications are available universally, usually bundled with infrastructure software at no cost. This sets the expectation. If somebody buys a computer, it's able to process XML. What's meant by XML is clear: it's defined in the W3C's recommendations.

At the foundation of the universal integration back plane we need an open specification for reliable, secure messaging that everyone builds into the computing infrastructure at no additional cost. Like HTTP and XML – open and fully interoperable.

Most of the needed specifications are available in draft form from the W3C XML Protocols activity. New Working Drafts were published June 26:

- XML Protocol (XMLP) Requirements
- SOAP version 1.2, Part 1: Messaging Framework
- SOAP version 1.2, Part 2: Adjuncts
- SOAP version 1.2, Usage Scenarios

Additional specifications that take Web services to the next level are becoming available – for example, the OASIS WS-Security Technical Committee, announced on June 27. At Ford Motor Company we believe that these and other "WS-protocol" specifications will define the core technologies for effective, industry-standard communication between corporations and applications over the Internet.

Several past efforts showed some promise of delivering the communications layer we need for the universal integration back plane: RosettaNet Implementation Framework, BizTalk Framework, and ebXML Messaging, plus many specifications developed by various vertical industry associations. Web services differs from previous efforts in several important respects:

- The effort has the full support of the entire software industry.
- The work is being done within strong global standards organizations and consortia.
- There is commitment to ensuring through implementation experience that these specifications deliver the intended results.

Web services have full commitment from essentially all software vendors. IBM, Microsoft, Oracle, SAP, and Sun, among others, have all stated that they will fully support the Web services specifications. Their representatives say so at conferences and meetings. It's in their

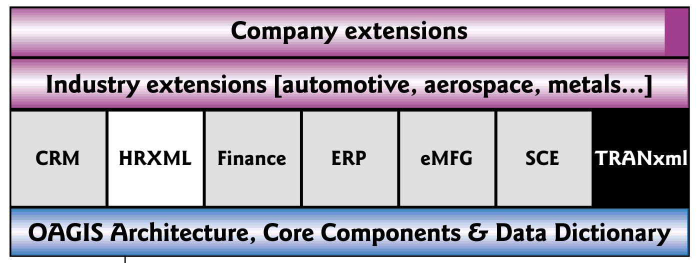

Company extensions
Industry extensions [automotive, aerospace, metals...]
CRM
HRXML
Finance
ERP
eMFG
SCE
TRANxml
OAGIS Architecture, Core Components & Data Dictionary

FIGURE 1 | Mechanisms for creating horizontal and vertical extensions

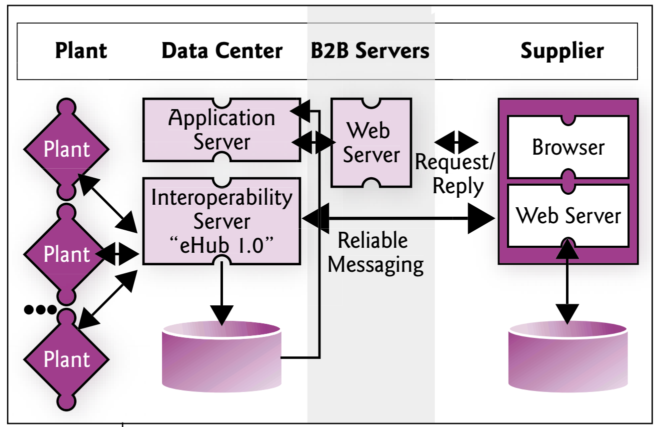


FIGURE 2 | eHub 1.0

product literature and press releases. These companies intend to ensure that all their Web services implementations will interoperate with other vendors' Web services products. Everyone realizes that multiplatform interoperability is essential to the Web services vision and promise.

The home for the core of this work is the W3C, which has an outstanding track record of producing high-quality specifications that in many cases are universally adopted. The W3C recommendations define the fundamental technology of the World Wide Web. Other Web services specifications (WS-Security, UDDI) are going to OASIS, which already is home to some related specifications (ebXML, SAML). It's encouraging that these specifications are in W3C and OASIS, not in several competing, locale-specific, or vertical industry–specific standards organizations.

Despite the best of intentions, people can interpret even the most effective standards and specifications differently and therefore build incompatible implementations. Experience from production deployment has to be fed back into improving the specifications and the software that implements them. To address this, the software vendors have formed WS-I, and have invited other companies to join them. Ford Motor Company has committed to become a member. Independent of the WS-I activities, the W3C XML Protocol Working Group will not advance the SOAP specifications on their way to becoming W3C Recommendations until two conditions (among others) are met: that (1) "sufficient reports of implementation experience have been gathered to demonstrate that SOAP processors based on the specification are implementable and have compatible behavior," and that (2) "an implementation report shows that there are at least two different and interoperable implementations of every mandatory and optional feature."

Given the groups formed in W3C, OASIS, and WS-I, and the widespread support in the software industry, the chances for success this time look good. We'll have an open specification for reliable, secure messaging that everyone supports.

In the past people have been concerned about the lack of Web services specifications covering security and reliability. At Ford Motor Company we find that XML and earlier versions of SOAP are excellent technologies to use over reliable, secure transport. We've used them for years. But when we use them over nonreliable, nonsecure transport, we're very careful what kind of business processes and data we use them for. To get the degree of reliability our users need, we build store-and-forward, retry mechanisms, roughly along the lines of the reliable messaging sections of the BizTalk Framework and ebXML Messaging specifications.

Wireless networks and the Internet are neither secure nor reliable. Nor are corporate wide-area networks. We have incidents recorded in our maintenance logs of servers going down and network connectivity being flooded or unavailable. Because we implemented our Web services to be tolerant of these problems, we still get our SOAP requests through, once connectivity and service are restored.

We're very interested in using Web services over imperfect transport and server infrastructures, such as the Internet. To do this we need to secure the XML documents themselves. When there is the possibility of unplanned downtime in the networks and server infrastructure, we need to be able to retry the Web services requests. It would also help if the Web services infrastructure were intelligent enough to automatically stop transmitting when service becomes unavailable and resume when service is restored. We also need to be able to use Web services in both a synchronous (request-response) and asynchronous (event-driven) mode – and the latter requires a well-defined receipting mechanism to guarantee and subsequently validate that the message got through.

The new Web services drafts include more than the combination of XML, SOAP, UDDI, and WSDL formerly contained. The WS-Security specification is a good addition, and will improve as it moves through the OASIS Technical Committee process. Sun announced on June 27 that they, along with the other OASIS members, will join the specification's original authors – IBM, Microsoft, and VeriSign – in developing and supporting it.

Some items haven't been addressed yet. The ebXML Messaging Service specification includes a Reliable Messaging option, using the same message store-and-forward algorithm that's in BizTalk Framework 2.0. The W3C XML Protocol Requirements stop short of defining reliable messaging, although key ingredients are provided. Usage Scenario S5, "Request with acknowledgement," covers it in part, but includes a note: "This scenario does not imply that reliable message delivery will be supported by the XMLP core specification."

BizTalk Framework 2.0 and the Reliable Messaging part of ebXML Messaging Services are useful tactical solutions for current implementation. There are draft Web services specifications under review, including IBM's proposal for reliable HTTP (HTTPR). HTTPR has the advantage of being able to support all types of Web services interaction rather than a special category of reliable messaging. Other protocol-agnostic proposals also exist.

The chances are good that we'll see the gaps filled. Many previously developed specifications will converge around Web services protocols. Ford Motor Company is pleased to join with other end-user companies that pursue XML excellence in making our views known to the IT industry. It helps if we customers are clear about what we want. I think it's reasonable to ask for:

- Infrastructure for reliable, secure XML-over-HTTP (and other protocols) Web services
- Secure, reliable delivery offered as a standard feature of the infrastructure platforms (J2EE, .NET, whatever else makes sense)
- Commoditized, standards-compliant implementations of supporting services – security, routing, transformation, logging, metering, billing, process aggregation, and transaction support
- Full interoperability across all implementations
- Compliance with specifications published as open standards or recommendations

XML Implementation at Ford Motor Company

Ford is among many companies implementing XML, SOAP, and Web services technology so as to build better applications and better integrate them. Our first production implementations used SOAP to enable us to locate Web applications outside our firewall that access information held in databases inside the firewall. We found that SOAP gave us a much more manageable and secure method for communicating over the network and across the firewall than other technologies for connecting application servers to databases. Use of XML between applications and trading partners came a bit later.

For a recent project we wanted to enable our suppliers and logistics providers to know in near real time when we've used parts they bring to our plants. This required an event-driven paradigm rather than synchronous request-reply use of SOAP. We wanted to publish information generated by part consumption events to our trading partners and to

PolarLake

www.polarlake.com

applications located in our central data centers. To support this project we built infrastructure (shown in Figure 2) that we called "eHub 1.0."

This project by itself justified acquiring and putting in place the infrastructure. We and the IT vendor who provided the interoperability server product spent extensive effort in testing and improving it so that it's reliable and can support all plants and all parts. The project meets our current stringent requirements for financially justifying investments. This first integration project's success demonstrated the value of near real-time event-driven communication and the affordability of XML technology.

The eHub 1.0 infrastructure is reusable. During the time it took to implement our first application, we designed and built a second one (production tracking to support outbound logistics). We launched this application on the eHub 1.0 infrastructure as well. Since our launch of these two applications in May 2001, we've added applications supporting purchasing and dealer communications. Using XML and OAGIS standards accelerates projects and reduces risk by enabling us to rapidly stabilize project requirements. Both the infrastructure and the information flows themselves are reusable. We reuse the flow of production tracking information in two additional applications that weren't originally planned.

XML and Web services are rapidly evolving technologies. We've built an eHub 2.0 that makes use of newer standards and increases available capacity to enterprise levels.

Next Steps

One of the inhibitors of more widespread use of XML and Web services for integration is immaturity of the standards. What can we do, as a software industry, as users of information technology, as XML experts, to help this along? Here are suggestions. People are probably doing these things, but it doesn't hurt to restate the obvious.

- ***If you have expertise:*** Comment on the W3C and OASIS Web services drafts. Join the WS-I and the Open Application Group consortia. Participate in the subcommittees and activities.
- ***W3C and OASIS Web services committees:*** Take previous specifications and implementation experience, including ebXML, fully into account. Understand that users of the specifications need to follow a migration path from current state to a coherent, industry-wide set of fully supported protocols.
- ***OASIS ebXML Technical Committees:*** Fully support and align with the Web services protocols efforts.
- ***OASIS:*** Work closely with W3C to facilitate a migration path between the ebXML and Web services protocols initiatives, consistent with the W3C Web services architecture.
- ***End-user businesses and organizations:*** Build your piece of the universal XML integration back plane we've been talking about. Ask for open XML document definitions in the interfaces of products you buy, rather than proprietary interfaces. Use communication technology that's compliant with SOAP and the emerging Web services specifications. Software companies are much more motivated by the marketplace than by standards groups and consortia.

If you manage a company or organization that's working on ebXML, Web services, and other XML projects, please make sure that the people working on your various projects align their activities. Consider operating all those projects out of a single department.

Don't promote a continuing feud or debate on the essential technologies. For example, if part of the world does a Web services–reliable messaging extension of SOAP and the rest does ebXML Messaging, and the two don't interoperate, everybody loses.

The return on investment for our XML projects is much more compelling for internal projects at Ford Motor Company than for B2B projects, where we already have X12 or EDIFACT EDI in place. But when we do see value in aiming our Web services requests at URIs outside our firewall, we don't want to be required to change to some other set of standards and technologies. We want to use the same technology that we're using inside the enterprise.

What about UDDI, WSDL, UBL, ebXML Registry and Repository, BPSS, WSFL, UN/CEFACT core components, and so on? Of four "core" Web services specifications (XML, SOAP, UDDI, WSDL), we've used XML and earlier versions of SOAP at Ford Motor Company for years. We've done preliminary work with UDDI and WSDL. At current scale, manual procedures for things like registry, service description, and collaborative partner agreements are workable, but these and additional technologies will be useful in the future as the world's universal integration back plane grows in scale.

With its support of XML, OAGIS, and Web services protocols, the software industry is about to offer its customers the elements (Communications, Syntax, Meaning, Process) of a universal integration back plane built on Internet technology that every application and computing device plugs into. These elements will come with the applications and infrastructure software we buy. They will ultimately even come with open source software. There are already open source implementations of XML parsers, and there is at least one open source ERP business software package that implements OAGIS. VeriSign has contributed an open source reference implementation of WS-Security.

The Challenge of XML Excellence

To get the most from XML and Web services technology, there are things we need to do as end-user IT organizations:

1. Use the overall structure and integration best practices of the OAGIS. Learn them, extend them, and engage with the OAG members in improving the specification.
2. Use and support the relevant XML and Web services standards. Feed back implementation experience and requirements to the groups that are developing and supporting them.
3. Manifest XML excellence in a focused solution that delivers business value to the company. Build a reusable infrastructure like Ford Motor Company's eHub through which people can see the standards and best practices for integration at work, and interfaces can be reused across integration efforts.
4. Drive use and compliance internally and externally to the organization. This technology grows in value the more broadly it's used – there's a strong "critical mass" effect. Use by only one project, one department, or one enterprise is of limited value.
5. Plan for broad use of event-based messaging, request-response Web services, and electronic collaboration. At this time, most attention must be paid to the essential standards: W3C Web Services Protocols and WS-Security. They're needed now to enable widespread adoption and to provide the base on which to grow. Soon we'll need support for the more complex interactions. We must be prepared to drive, develop, and support the new technologies such as WSDL and UDDI, and we need to be ready to migrate the infrastructure that supports our XML solutions.

The challenge now is to specify, develop, and implement these essential technologies in a way that enables all to participate and that delivers clear financial value from every implementation. We have to build a solid base, and at the same time think creatively. This technology will succeed if it's good for business. Everyone's business.

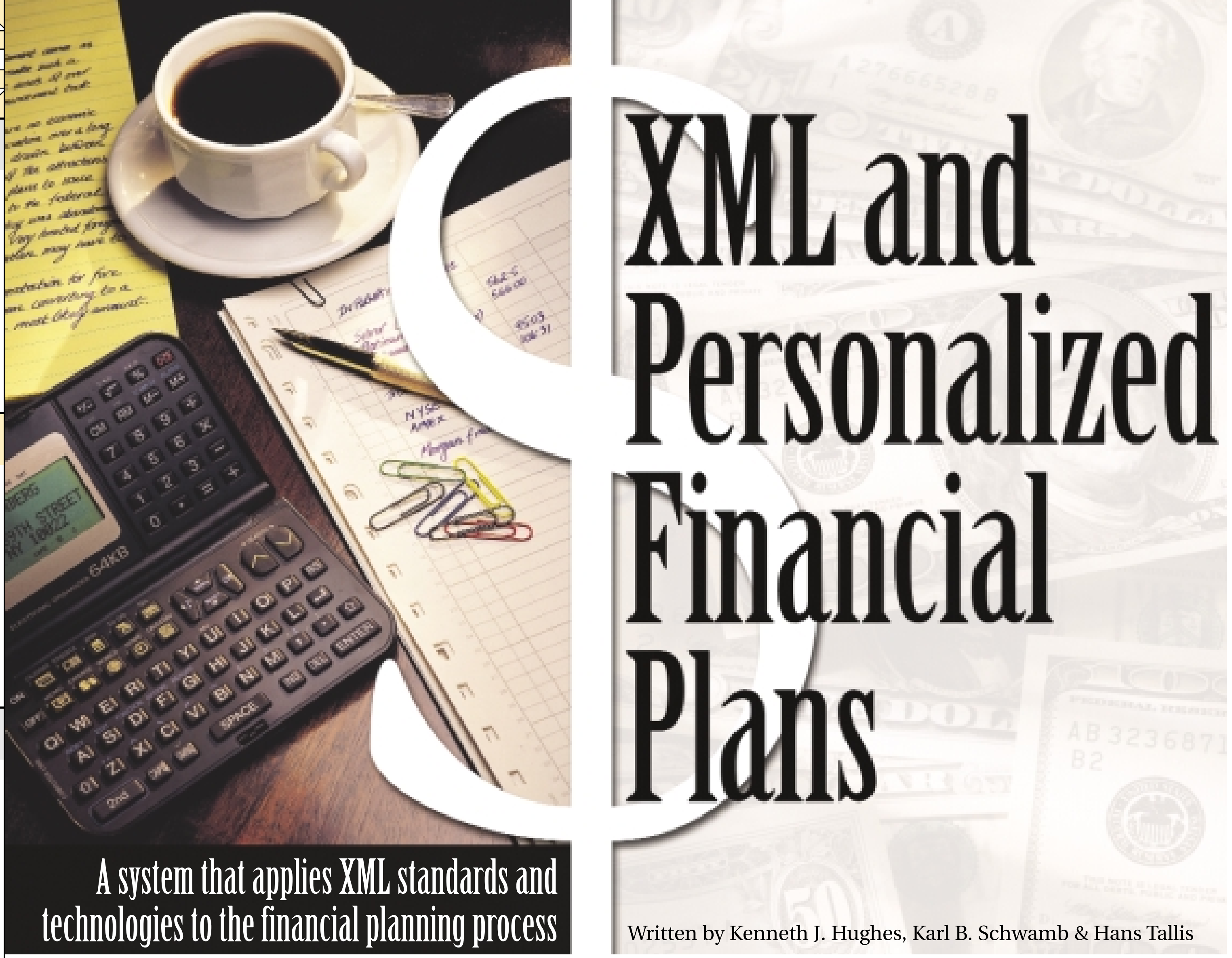
ABOUT THE AUTHOR

Tim Thomasma, a senior technical specialist in Ford Motor Company's Infrastructure Architecture group, is responsible for the architecture of Ford's plant floor systems, where data is collected from a variety of manufacturing devices and integrated into Ford's information systems. He has extensive background in application development, manufacturing systems, and simulation. Previously Tim was senior consultant in enterprise Java services at IBM Global Services. He serves as cochair of the Automotive Industry Action Group's XML/EDI Work Group.

TTHOMASM@FORD.COM

Mongoose

www.portalstudi.com



XML and Personalized Financial Plans

A system that applies XML standards and technologies to the financial planning process

Written by Kenneth J. Hughes, Karl B. Schwamb & Hans Tallis

The publishing industry has long used markup languages in the production of publications to a general readership. Now XML greatly facilitates the design of publishing systems whose capabilities include not only control of formatting, but also control of content selection according to the needs of the individual reader.

Customers of financial services firms don't need to settle for advice written to a general readership; they can receive plans so personalized and uniquely suited to them that they'll feel a team of experts conferred regarding their situations and wrote specific, 200-page recommendations for planning their financial futures.

Through the application of XML standards and technologies to the financial planning process, the system described in this article automatically produces highly personalized financial plans. The article emphasizes the role XML plays in such a financial planning system, and presents techniques that are applicable to any document personalization system that must operate and evolve in a production setting.

Specifically, we discuss the use of:

- *XMLC* in the Web presentation layer for managing the data that drives personalization
- *XSLT* for assembling personalized documents
- *DocBook* for representing personalized documents
- *Meta-XSLT* for generating user-readable documentation of the personalization process

Background and Requirements

We receive simple personalized documents nearly every time we sort the day's mail – utility company invoices and brokerage account reports are familiar examples. Basic document generation packages that support the production of these documents provide certain simple features:

- Inclusion of parameterized numeric and string data
- Limited use of custom graphics, such as an asset-allocation pie chart or savings-goal thermometer
- Limited conditional text logic, typically at the "paragraph" level of granularity
- Performance supporting upwards of 10 million reports per month

What follows describes an XML-based document production system that extends the functional and technical capabilities of those simple packages. The features include:

- *Flexible structural control,* including document organization with hyperlinks
- *Production-quality print graphic format* (scalable vector format) and efficient Web graphic format (raster)
- *Full layout control capabilities* (e.g., headers/footers, tables, flowing text)
- *Multiple output formats* (PDF, RTF, XHTML, ASCII text)
- *High-visibility and direct-manipulation control* of document content; clients review and directly edit the document generation templates
- *Unlimited complexity* supported in the logic of text inclusion, allowing arbitrary Boolean expressions to control conditional text inclusion. Conditional text may be as fine grained as the subword level (e.g., verb tense endings) or as coarse grained as entire chapters
- *Platform portability* through Java implementation
- *Standards-based protocols and languages*

These features allow us to meet the demanding needs of high-volume financial plan generation. Comprehensive financial plans that cover areas such as investment, retirement, and estate planning, while simultaneously addressing tax implications, insurance sufficiency, and risk tolerance, can be over 200 pages. The input information for such plans numbers in the hundreds of data items and spans areas such as income, expenses, family educational goals, and retirement lifestyle.

Subtle changes in data values can have ripple effects throughout the plan, affecting numerical results, item cardinalities, and even the suitability of entire chapters. The directed alteration of input data in "what-if" scenarios requires rapid turnaround. Geographical distribution is necessary for user and customer convenience, input data collection, review of generated reports, and support of international users. Once the financial data has been analyzed and alternatives have been considered, high-quality financial plan documents are printed and delivered to customers.

To produce a high-quality, branded presentation for the Web site supporting the effort, we used a separate Web design firm to develop the look-and-feel of the Web pages. The site was designed in several iterations, requiring several updates for the programming team. This cooperative effort required careful separation of the work of the design team versus the programming team.

The group of certified financial planners and legal counsel advising the effort is even more critical to its success. This team must carefully monitor the wording of the financial reports to ensure they are correct and complete over the many combinations that can be generated by the system. It's important for such a team to have good visibility into the document generation process.

Architectural Overview

It was decided early on in the project to utilize a Web-based interface to the application. This addresses the need to (1) handle a geographically dispersed and mobile user community, (2) support a variety of user platforms, and (3) comply with open standards important to IT departments. To address the need for multiple user roles (such as field users, managers, administrators, reporters), the Web interface supports many configurations. The requirement that a separate organization develop the look-and-feel of the application meant the presentation specification and its programmatic implementation should be as loosely coupled as possible.

These requirements were satisfied quite well using the XMLC tool, which produced a much cleaner separation between presentation and implementation than server page technologies. Figure 1 illustrates how page designers produced validated XHTML that Java developers could use in the application server without manual modification. (This process is described in more detail in the next section.)

Once the customer data is captured, it's analyzed and used to produce a customized financial plan. The customer data is stored in a conventional relational database – it's extracted by custom Java code that performs numerous calculations to analyze the customer's situation and produce tailored recommendations. The analysis results are represented in a custom XML document that's completely data-centric. XSLT stylesheets are employed in a "pull" style to conditionally include text fragments and graphics. The target representation of the result is in DocBook, a rich XML application for representing books and articles. This approach to generating a personalized financial plan in a generic format is shown in Figure 2. (The process is elaborated in a later section.)

Once a financial plan has been generated in the intermediate DocBook form, the standard DocBook XML stylesheets are used to produce output in XHTML (for the Web) or RTF (for postprocess manual customization). PDF output, for high-quality print production, is also produced with the assistance of custom XSL-FO stylesheets. The fact that source code for these standard stylesheets is available is an important risk-reduction factor; they can be customized to produce the desired output styling, if necessary. Since they were included with DocBook, this greatly reduced the amount of custom code needed to produce high-quality output. Figure 3 illustrates the use of the standard DocBook stylesheets. (Further information on this approach is presented later.)

When and how text fragments are included in a financial plan is a highly sensitive issue. Hence it's important that CPAs and legal advisors have visibility into the process. This is achieved by automatically producing documentation from the stylesheets used to generate personalized financial plans. A meta-XSLT stylesheet is used to translate the financial plan generation stylesheets into readable documentation. This automatic process ensures that the documentation always reflects the current state of financial plan generation (see Figure 4). Once the documentation in DocBook is produced, the same process illustrated in Figure 3 is used to present the documentation to the content and legal experts of the business sponsor. (This process is described in more detail later.)

The processes illustrated in the figures are carefully coordinated to reduce user wait time. The online Web interaction that manages customer data is designed to operate on current information in the database. A queue of user requests feeds separate processes that implement Figures 2 and 3. When the user requests have been completed, generated documents can be retrieved for viewing or sent for hard-copy print production. This allows for rapid Web interaction through a pool of machines configured for high user interaction. A separate pool of machines consumes and satisfies requests for document production. Since the document production task is batch-oriented, those machines utilize a different configuration. The process illustrated in Figure 4 is an offline process performed once per build for the entire application.

Managing Personal Financial Data

To manage customers' personal financial data, financial analysts across the country need convenient access to centralized financial records. All of the usual advantages of Web-based systems (user familiarity with the interface, universal availability of browsers, ubiquity of network infrastructure, etc.) applied to this system. One particularly salient characteristic is the amount of information that must be managed for each customer. A team of specialists in user interface and graphic design sketched nearly 100 form-based pages for the management of customer data. Coordination of the initial development and subsequent maintenance of these pages between graphic design and programming teams was greatly facilitated by the development of a unified framework for page management based on XMLC.

XMLC compiles XHTML into Java code that can be manipulated programmatically at page presentation time to include dynamic information. Graphic designers can create page layouts using their favorite editing systems, preview their work in standard Web browsers, and validate the XHTML encoding using an XML validation tool. Programmers can independently use XMLC to compile the completed pages into a Java-based representation that can be manipulated at page presentation time via a standard DOM-based API.

XMLC's separation of page-layout from back-end development provided substantial benefit to the project. Additionally, because there are so many pages to control and XMLC provided complete presentation-time control over each individual page, further gains were achieved by identifying common page population and database updating code that could be factored out across the ensemble of pages. This idea was taken to the extreme of writing an interpreter, called PageManager, that handles all page population and database updating.

PageManager uses the XMLC-generated DOMs of each page to examine the IDs placed on elements destined to receive dynamic content. Each ID is built from a simple grammar that is expressive enough to represent the customer's financial data in terms of scalars, lists, and constraints. PageManager interprets the IDs, uses reflection against Java-based domain objects (backed by an Oracle or DB2 database) to retrieve dynamic data, updates the DOM representation of the page, and writes out the XHTML representation at page presentation time. PageManager updates DOM representation to populate forms, label buttons and column headings, or add rows to tables, for example. It also writes "name" attributes using a special grammar that allows PageManager to update the Java-based domain objects when the page is returned after user editing.

While a full description of PageManager's ID and name grammars is beyond the scope of this article, an example will serve to highlight the basic operation. The following XHTML tag is a typical prototype row adorned with a PageManager ID that will cause the row to be repeated for each current customer asset with a CategoryUID equal to INVESTMENT_ASSET:

```
<tr id=":Profile:Asset.CategoryUID-INVESTMENT_ASSET">
```

Within such a prototype row might be an <input type="text"/> tag with an ID that directs PageManager to populate the control with the name of the institution related to the asset:

```
id=":Profile:Asset.CategoryUID-INVESTMENT_ASSET.InstitutionName"
```

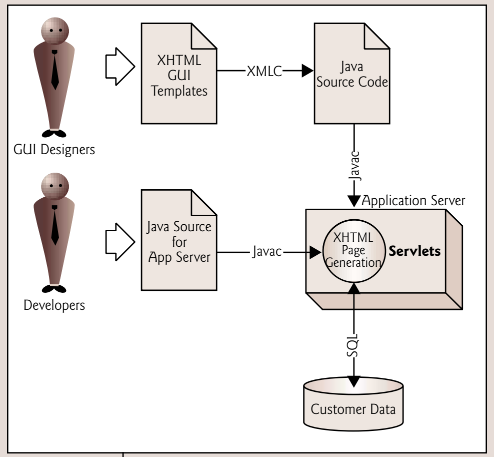


FIGURE 1 | Managing personal financial data

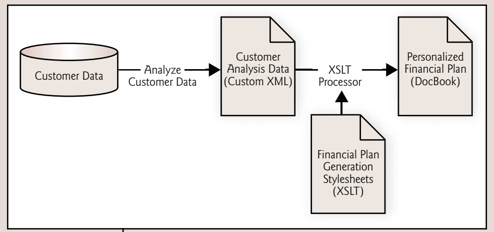

Customer Data
Analyze Customer Data
Customer Analysis Data (Custom XML)
XSLT Processor
Personalized Financial Plan (DocBook)
Financial Plan Generation Stylesheets (XSLT)

FIGURE 2 | Controlling plan content via XSLT

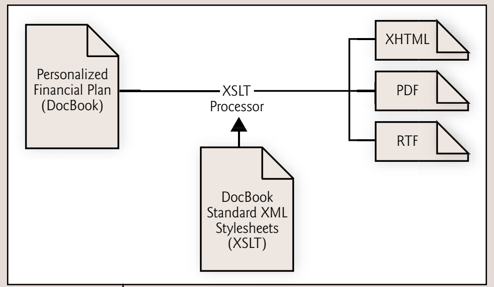

Personalized Financial Plan (DocBook)
XSLT Processor
XHTML
PDF
RTF
DocBook Standard XML Stylesheets (XSLT)

FIGURE 3 | Using DocBook to represent financial plans

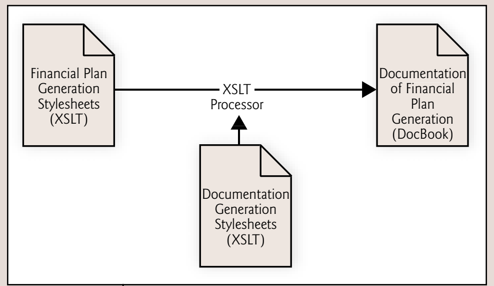

Financial Plan Generation Stylesheets (XSLT)
XSLT Processor
Documentation of Financial Plan Generation (DocBook)
Documentation Generation Stylesheets (XSLT)

FIGURE 4 | Documenting plan content control via meta-XSLT

PageManager would automatically assign a "name" attribute for each such input so that during form submission it can readily identify value originations and update the database appropriately:

```
name="Asset.UID-12345.InstitutionName"
```

Similar ID and name attributes guided the population of other controls, the updating of changed fields to the database, and the navigation to detail pages associated with data in the dynamically generated rows.

Controlling Plan Contents via XSLT

Financial plan personalization is the centerpiece of the system. The Java-based (and Oracle- or DB2-backed) domain objects mentioned in the "Managing Personal Financial Data" section represent all of the financial data relevant to a particular customer. These domain objects are used to perform various calculations and are then serialized to XML, which in turn serves as the input to an XSLT-based document generation process that produces the personalized financial plan in DocBook format. The DocBook-based plan is then processed to produce customer-readable financial plans in XHTML, PDF, and RTF (as described in the following section).

A pull-based XSLT processing model is used to produce the plans. XSLT templates conditionally include DocBook elements and financial planning content according to XPath conditionals against the XML representation of the Java domain objects and calculation results. Sections, paragraphs, sentences, words, subwords, and punctuation are included or excluded according to the financial planning knowledge applied against the customer's financial data and goals.

Using DocBook to Represent Plans

Financial plans are represented in DocBook because of the maturity and flexibility of the tools for processing that format. We were able to produce immediately results using standard DSSSL stylesheets for XHTML, RTF, and PDF. The XHTML produced was of satisfactory quality; RTF required some DSSSL customization to get acceptable results; PDF output was very rough, especially in the area of table formatting. We chose the DSSSL approach initially because the XHTML and RTF results were good, and the PDF results appeared to be close. The XSL stylesheets were relatively immature, as were the XSL-FO processors that were needed to produce PDF output.

However, a change in sponsor requirements led to a deemphasis of the need for RTF and a focus on the need to customize the PDF output style. The complexity of the DSSSL/OpenJade pipeline to PDF (DocBook->TeX->PS->PDF) frustrated our efforts to get clean PDF output. For example, finding how to fix footnote-formatting problems within tables along that pipeline proved to be daunting. When faced with the additional need to provide extensive stylistic customizations, we had little confidence in our ability to cajole the DSSSL-TeX-PDF pipeline into successfully producing the requested effects.

The purely XSL alternative to the DSSSL approach was still very immature. However, by working closely with the vendor of one of the leading XSL-FO processors (XEP, by RenderX), we eliminated the critical obstacles and proceeded to customize the default DocBook stylesheets in XSL to achieve the effects requested, such as:

- Scalable, vector-based EPS graphics for charts, graphs
- Tables involving cell spanning, proportional column widths, column alignment
- Customized styling for section headings and footnotes
- Graphics (logos and other stylistic designs) in headers, footers, margins
- Table of contents (TOC) customizations, table and figure TOCs removed, first chapter moved before TOC, etc.

The elegance of the XSL method of styling output is noteworthy. Mapping from an XML document to other XML documents (XHTML or XSL-FO) to produce output in the required styles using XSLT templates is a natural and effective way to specify style. DocBook's default XSL stylesheets provided a handy default that could be customized effectively by overriding selected XSLT templates.

Documenting Plan Content Control via Meta-XSLT

The pull-based XSLT processing model usually has the advantage of being relatively easy to read because the statements of conditional processing are embedded within a context of the targeted document. Even so, nontechnical people are quickly distracted, or worse, confused, when confronted with strange-looking XPath and other XSLT constructs, especially when embedded within DocBook tagging, which itself is unfamiliar to them. Furthermore, the high degree of personalization described in the previous section makes for such a high control/content ratio that only the most die-hard XSLT jocks like looking at the massively conditional templates.

Our solution to the problem of how to produce a readable specification of the conditional processing embodied in the XSLT templates was, perhaps surprisingly, to write more XSLT templates. The second set of templates forms a transformation from the original, difficult-to-read set of templates in XSLT to a rendition in DocBook that uses English-based descriptions of the conditional processing. This easily read specification in DocBook is then processed by the same stylesheets used to customize individual financial plans, so the business analysts and legal advisors can review the conditional text processing in context and in the exact style of the final product.

For more details on how a small number of push-based XSLT templates produces documentation on a very large set of pull-based XSLT templates in terms that a business analyst could easily read, see our "Document XSLT Automatically" at www.sys-con.com/xml/article.cfm?id=419, a companion article in the June 2002 issue (Vol. 3, issue 6) of *XML-Journal*.

Discussion and Conclusions

The approach described above has produced the most advanced financial planning document pipeline, in large part because of the leverage of XML technology. While the project requirements were numerous and demanded a great deal of flexibility, XML was up to the task.

Despite the success of the above approach, there are a few areas for improvement. It would have been helpful to have a single XML format for defining analysis calculations that could be used both to present these calculations to the project sponsors and to generate implementation code. These calculations are described in standard industry publications.

While MathML holds promise in that it can describe the presentation of calculations and be incorporated into DocBook, we know of no way to generate implementation code from this MathML markup. The addition of such a capability would, of course, provide visibility into a large requirements area that currently requires a great deal of manual effort.

Graphics were a challenge due to the need to support multiple formats since each document presentation required a different graphic format. Multiple graphic formats were produced in the document generation pipeline since the presentation output was not known until the end of the process. It would be much better if SVG were natively supported in browsers, Microsoft Word, and PDF.

Another area for improvement is the representation and presentation of business rules. Some key areas of financial analysis are best represented by rules that can be executed within a rule-based system. While several rule standards have been proposed, such as SRML, RuleML, and BRML, none have the status of an approved standard, and no path exists for documenting the rules.

Taking a wider look at the financial planning landscape, it's clear that once a plan is produced, customers prefer to have their plans monitored as they make updates to their financial situation. Many of the XML standards that already exist for trading and exchanging financial transaction data, such as IFX, could be used to update customer data over time and to perform plan monitoring. The monitoring activity can be used to make automatic portfolio adjustments, notify customers of deviations from plan goals, and cross-sell and up-sell financial instruments.

References

- *XMLC:* http://xmlc.enhydra.org
- *DocBook:* www.docbook.org
- *MathML:* www.w3.org/TR/REC-MathML
- *DocBook:* www.oasis-open.org/docbook/xml/mathml/index.shtml
- *SVG:* www.w3.org/Graphics/SVG/Overview.htm8
- *SRML:* http://xml.coverpages.org/srml.html
- *RuleML:* www.dfki.uni-kl.de/ruleml
- *BRML:* http://xml.coverpages.org/brml.html
- *IFX:* www.ifxforum.org

KJH@ENTEL.COM

KBS@COLONNADESOFTWARE.COM

HCT@ERS.COM

ABOUT THE AUTHORS

Kenneth J. Hughes is president of Entelechy Corporation (www.entel.com), a firm that specializes in XML.

Karl B. Schwamb is president of Colonnade Software (www.colonnadesoftware.com), a firm that specializes in distributed software systems.

Hans Tallis is a founder of ExpLore Reasoning Systems (www.ers.com), an automated financial planning consultancy.

When asked to join the OAGIS modernization project (OAGIS 8), I leapt at the chance. Here were two renowned specifications just waiting to get acquainted:

- Open Applications Group's OAGIS, the solid, proven Integration Specification, an early XML application (1998) with a lot of miles on it
- W3C's XML Schema Recommendation (hereinafter "Schema"), a sophisticated new metamodeling chassis, ready to be road tested

OAGIS 8

Written by
Mark Feblowitz

Practical integration meets XML Schema

The outcome exceeded expectations: OAGIS's established and widely used family of horizontally focused, DTD-encoded interchange messages was updated, eliminating major usability issues inherent in a DTD-based implementation of this scale. At the same time, Schema enabled the horizontal OAGIS specification to employ an extensible architecture, OAGIS 8 Overlay Extensibility, to address the specific needs of vertical industries. Vertical market fit versus broad horizontal reusability has long been a sticking point between competing integration standards – Overlay Extensibility enabled an approach that leverages the strengths of both models.

Developing a specification that promotes both horizontal coverage and vertical specialization wouldn't have been possible without Schema's advanced capabilities. However, the experience also opened our eyes to some challenging aspects of developing a usable, practical XML Schema solution to a complex real-world problem.

This article describes the Open Applications Group Integration Specification, discusses the enhancements made possible by rearchitecting to Schema, and explores the challenging aspects of applying current Schema technology. Despite those challenges, OAGI architects were able to work with Schema to craft a new OAGIS that sustains proven strengths and adds desirable and innovative features, most notably, Overlay Extensibility.

What Is OAGIS?

- ***OAGIS is an Integration Specification:*** A technology-neutral means of integrating enterprise applications and of engaging in e-business transactions. We live in a world where integration is a necessity, not a luxury: enterprises face EAI scenarios daily – within their enterprise, with partnered enterprises, and, increasingly, with merger partners; and they steadily offer and consume more e-business services. Having a technology-neutral reference specification provides the clear benefit of minimizing pairwise integrations.

  For OAGIS, Integration Scenarios (see Figure 1) are the basis for integration message design, establishing the context of messages that accomplish the integration, whether B2B or EAI. These scenarios are reusable designs that can be modified to meet specific business process needs.
- ***OAGIS is the BOD:*** The Business Object Document, OAGIS's atomic transactional interchange message, is the structured XML message interchanged between applications, either intra- or interenterprise. BODs (e.g., ProcessInvoice, ChangePurchaseOrder) represent applications' integration APIs and/or the enterprise's e-business service interactions. The BOD defines, among other things, a Noun, the business object (Invoice, PurchaseOrder,...) that is the subject of the interchange; and a Verb, the operation (Add, Change, Process, Cancel,...) to be applied to the Noun. Benefits of using the BOD message architecture include consistency of architecture, message, and dictionary; high levels of reuse across messages; rapid development; and a smaller learning curve for users and developers.
- ***OAGIS is a canonical language:*** OAGIS provides a horizontal set of messages to address a broad spectrum of typical e-business and EAI interchange needs composed from a large set of core components.
  A canonical, general core establishes a strong common base. Yet no business or industry consortium would feel that this alone could cover all of their interchange needs, nor could it adequately express their segment's specific vocabulary. Because of this…
- ***OAGIS is extensible:*** Vertically adaptable to specific industries and enterprises, core OAGIS can be nominally extended using UserArea Extensibility, a rudimentary means of plugging additional content into standard BODs. OAGIS can be more substantively extended using Overlay Extensibility, where specific, new vocabularies are built on core OAGIS by extending existing BODs and components and/or by creating new ones. Extension is better than building from scratch: building on an established base enables reuse.
- ***OAGIS is transport independent:*** It's not a communication protocol, nor is it tied to a specific messaging technology. Any message transport can deliver a BOD: HTTP, HTTPS, SMTP, SOAP, publish/subscribe models; both synchronous and asynchronous communication models are supported.
- ***OAGIS is member-developed:*** BOD existence and coverage are dictated primarily by member need (not solely by theory of relevance), informed by member-developed interchange scenarios.
- ***OAGIS is usable, practical, maintainable:*** OAGIS elides details and complexities of the underlying representation, important where XML Schema is considered. Messages are constructed from core components; better modularization leads to reusable BOD processing code.
- ***OAGIS is established, widely used:*** OAGIS debuted in 1996; the first XML release was in 1998, roughly concurrent with the XML 1.0 Recommendation. OAGIS has been deployed at hundreds of sites worldwide. Its business-focused, transport-independent message architecture has made it a good fit in Web services deployments.

  OAGIS 8, the first OAGIS version adapted to XML Schema, was released in April 2002. In its first 10 weeks OAGIS 8 was downloaded by more than 1,500 companies, organizations, educational institutions, and individuals from more than 30 countries, representing a broad spectrum of international interest.

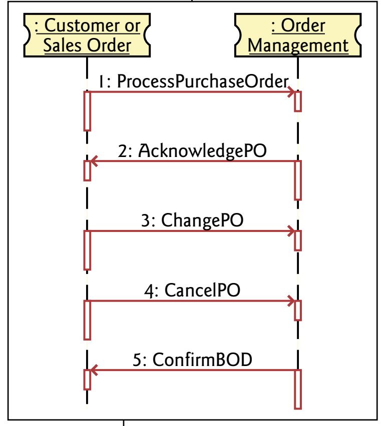


FIGURE 1 Purchase order interchange Integration Scenario

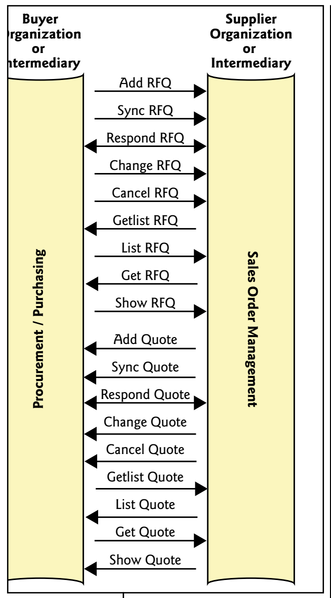


FIGURE 2 E-business Integration Scenario

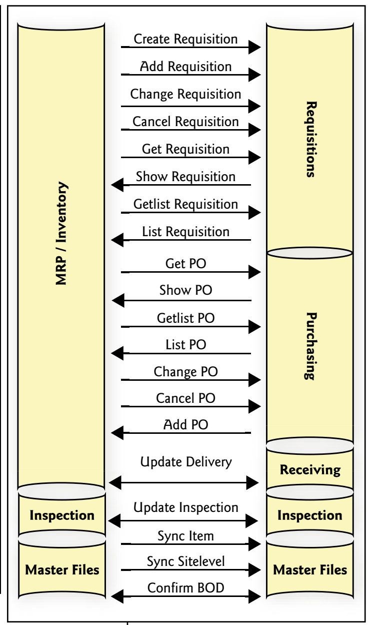


FIGURE 3 EAI Integration Scenario

The move to OAGIS 8

Although OAGIS use is widespread and increasing, members and users requested that the specification move to XML Schema, with its enhanced capabilities. Reasons for updating include:

- OAGIS stretched DTDs past their expressive limits, resulting in their being difficult to read and suffering from nondeterminism.
- DTDs' weakness in abstracting shared content contributed to inconsistency among BODs.
- DTDs' single, flat, all-global namespace led to longer-than-necessary names and prevented uniform naming of element children (rather than declare that every Order has an "ID," each had to be named uniquely, e.g., PRODORDID, SALESORDID).

OAGI's modernization goals for OAGIS included:

- Move to Schema but stay simple and usable.
- Retain and enhance OAGIS strengths.
- Take advantage of Schema's advanced features where appropriate.

OAGIS 8…

- Incorporates Schema's expanded type facilities to enhance both structural and content validation
- Uses abstraction to normalize shared content
- Adopts XML Namespaces
- Uses Schema extension capabilities to improve extensibility
- Eliminates designed-in nondeterminism

…all while remaining loyal to OAGIS's strengths: its usability, extensibility, and recognized transport-independent Verb/Noun message architecture.

In addition, OAGIS 8 introduces new features:

- Validated UserArea Extensibility
- Overlay Extensibility
- Adoption of XPath to simplify "Sync" and "Get" indicators

OAGIS 8 does not significantly alter the types of BOD messages or their information content.

OAGIS Basics

OAGIS is composed of roughly 200 BODs, applying ~20 Verbs to ~70 Nouns (not all Verb/Noun combinations are meaningful or relevant). Examples are shown in Table 1.

BOD design is a joint effort among interested OAGIS members, guided by the multiple, member-developed Integration Scenarios (see Figures 2 and 3) in which the BOD is expected to participate.

Given two or more interacting applications or enterprises, these scenarios help to elaborate the integration protocols that BOD interchange will enable, determining the number and type of BODs to be interchanged and the information to be carried by them. Since Integration Scenarios capture only an indicative subset of possible scenarios, they are neither XML encoded nor interchanged (OAGIS relies on technologies such as ebXML BPSS to capture coordination semantics).

Each BOD contains an ApplicationArea and a DataArea. The former carries application-specific (but not transport-related) content. The latter carries the Verb, with its operational data, and the Noun, the BOD's primary "business object" payload. A ProcessPurchaseOrder BOD would look like the following (with much more content), with the schema diagram in Figure 4.

```
<ProcessPurchaseOrder environment="Test" lang="en-US">
 <ApplicationArea>...</ApplicationArea>
 <DataArea>
  <Process acknowledge="Always"/>
  <PurchaseOrder>
   <Header>...</Header>
   <Line>...</Line>
   <Line>...</Line>
  </PurchaseOrder>
 </DataArea>
</ProcessPurchaseOrder>
```

A BOD carries one Verb and one or more Nouns of the same type. Several ProcessPurchaseOrder operations can be consolidated into a single ProcessPurchaseOrder BOD, with a single Process Verb and multiple PurchaseOrder Nouns. To keep the interface clean, one BOD cannot Process both a PurchaseOrder and a SalesOrder, nor can one BOD both Cancel one PurchaseOrder and Process another.

OAGIS Nouns are composed of a fixed combination of predefined core components: Fields, Compounds, Components. Fields equate to Schema simpleTypes, Compounds to complexTypes. Components are also complexTypes, but are special due to their extensibility, supporting both UserArea extensibility and Overlay extensibility.

BOD interchange would be meaningless without BOD processing. Like all XML interchange, a BOD is generated by the sending application, which serializes content from database tables or business objects, conformant to the BOD's respective schema. The BOD is consumed by the receiving application; it's typically validated against its schema and checked for adherence to BOD constraints (more on this later) and then mapped into tables or business objects. The application continues its processing, likely following up with subsequent BOD interchange.

BOD processing architectures vary, based on transaction loads, performance needs, application architectures, and the like. BODs can be validated and constraint-checked using either DOM- or SAX-based architectures. Validation can be disabled for performance purposes among stable, trusted interchange partners.

OAGIS Extensibility

Much of what's needed for typical business transactions is captured in core OAGIS. But needs differ; industries, organizations, companies conduct business according to their specialized vocabularies. OAGIS was designed with this in mind.

OAGIS 8 supports two forms of extensibility: UserArea Extensibility and Overlay Extensibility. When core OAGIS carries most of what's needed, an OAGIS user can add content to any BOD by populating one of the optional UserArea elements. When BOD extensions are numerous, or when additional BODs and/or components are needed, the OAGIS user can build Overlay Extensions, creating new vocabularies by building on core OAGIS components and BODs.

UserArea Extensibility

The UserArea, OAGIS's earliest form of extensibility, adds optional, open-content UserArea elements to every extensible OAGIS component, allowing any needed content to be added. The Charge component in Listing 1 has a UserArea that incorporates both an EffectivePeriod and a FairnessScore, neither of which was defined in the OAGIS Charge component.

OAGIS 8 builds on OAGIS's earlier USERAREAs by employing Schema's xs:any construct, and further tightens constraints on content via Schema's strict processing. Any new content that appears in a UserArea must be defined in some referenced schema, either within OAGIS or in some imported schema. This ensures that any BOD content is valid according to some predefined, accessible schema.

By convention, any non-OAGIS content must be imported from a non-OAGIS namespace. In the example in Listing 1 the Charge component contains a UserArea that incorporates the OAGIS EffectivePeriod element and the imported "myns:FairnessScore" element. The Charge component's UserArea element contains a declaration for the namespace "MyNamespace" and uses the schemaLocation to locate the schema for that namespace.

One drawback of UserArea extensibility is that there's no convenient way to restrict specific UserAreas to carry specific content – any defined content can appear in any UserArea.

UserArea extensibility is appropriate when the OAGIS base covers most of your needs – when you need only a little more content and don't mind having it segregated in a separate UserArea.

Overlay Extensibility

Among OAGIS's new features, Overlay Extensibility is most noteworthy. It provides a means of layering industry/vertical-specific vocabularies on OAGIS's canonical base without having to invoke any weighty, central standards evolution process.

Overlays can be layered on the OAGIS core (and upon each other) to accommodate specialized vocabularies: industry-specific, such as automotive, human resources; subindustries, such as auto parts, auto retailing; and company-specific extensions to adapt industry overlays to company-specific models and practices.

With Overlay Extensibility new layers are defined in their respective namespaces. Specialized BODs and components are defined by extending BODs from lower layers and/or by composing new BODs from a combination of existing, extended, and new

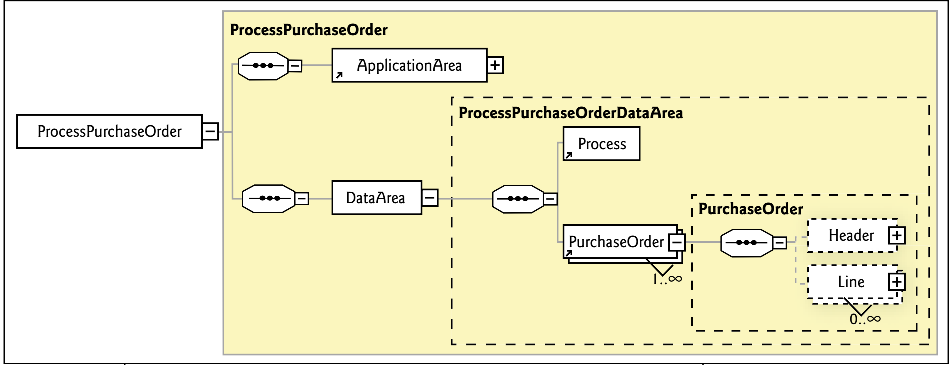


FIGURE 4 | ProcessPurchaseOrder BOD's schema definition

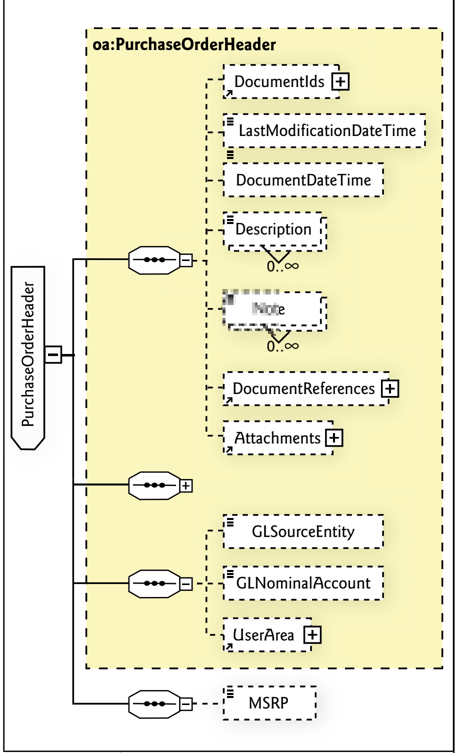


FIGURE 5 | Extending PurchaseOrderHeader components. To support this, OAGIS BODs – and most constituent parts – are overlay extensible.

To achieve Overlay Extensibility, OAGIS relies on three Schema mechanisms:

- Schema's incorporation of XML Namespaces
- complexType derivation by extension
- Substitution groups

XML Namespaces provides a clean mechanism for disambiguating potentially overlapping vocabularies; complexType derivation by extension provides a means of building on existing types any additional content that might be needed; the substitution group mechanism provides a means of plug-replacing an element with a related, extended element.

That last bit is the magic that makes Overlay Extensibility work. Without it the OAGIS user would incur the severe burden of trying to use Schema derivation of complex types by restriction to incorporate the extended content – and the ensuing intractable mess.

To demonstrate Overlay Extensibility, consider a fictitious example of an automotive industry group, AutoGroup. With an extensive, industry-specific vocabulary, they decide to build an AutoGroup overlay extension to OAGIS. They select a namespace, http://www.AutoGroup.org/oagis (with namespace prefix "ag"), and proceed to extend core OAGIS BODs and author new BODs.

Now consider their need to extend a PurchaseOrder, for example, to add an MSRP element to the Header. Since a PurchaseOrder's Header is an extensible component, they can use the existing PurchaseOrder BOD and simply add their MSRP element onto the Header (they can reuse existing PurchaseOrder processing code with little additional coding).

They extend the OAGIS oa:PurchaseOrderHeader complexType, deriving their own PurchaseOrderHeader complexType by extension, adding the MSRP element, and assigning it a type (since they're good citizens, they reuse the OAGIS type oa:Amount):

```
<xs:complexType name="PurchaseOrderHeader">
 <xs:complexContent>
  <xs:extension base="oa:PurchaseOrderHeader">
   <xs:sequence>
    <xs:element name="MSRP" type="oa:Amount" minOccurs="0"/>
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

Note that the extension defines only new content; other OAGIS content is "inherited."

The use of an IDE can simplify this derivation. Instead of being a schema-coding exercise, only a few gestures are needed to add element nodes to a Schema graph – for example, extending oa:PurchaseOrderHeader by adding a sequence node and an MSRP element node (see Figure 5).

AutoGroup then defines their Header to be in the substitution group for OAGIS's oa:Header:

```
<xs:element name="Header"
   type="ag:PurchaseOrderHeader"
   substitutionGroup="oa:Header"/>
```

Their Header is now a legal plug replacement for the OAGIS Header.

Finally, they define their own BODs that use the newly extended type; their new BOD, for example, ag:ProcessPurchaseOrder, now carries an ag:MSRP element in its Header. Any OAGIS enterprise that implements AutoGroup's overlay can now interchange an AutoGroup ProcessPurchaseOrder BOD.

If an MSRP element is later deemed "business-canonical," AutoGroup could request that it be incorporated into OAGIS; if approved and incorporated, MSRP could be dropped from AutoGroup's vocabulary.

If one of AutoGroup's member organizations would like to further tailor the vocabulary, they have a choice: get AutoGroup to incorporate their extensions, or build their own overlay – on both the AutoGroup overlay and OAGIS. Their decision rides on both the immediacy of their need and on the "fit" of their extension to AutoGroup's mission.

The Schema mechanisms underlying OAGIS extensibility are extremely complex. The recipe for constructing Overlay Extensions, though, is straightforward to apply (albeit by experienced integration professionals). One thing is clear: it's much easier to apply OAGIS extensibility conventions than it is to search the Schema Rec for a workable extension mechanism.

OAGIS 8 Representation Architecture

The overall representation style is a hybrid of OO single inheritance plus assembly of components. Single inheritance helps to make uniform the shared content of a dominant type. For example, all things of type Order share much common content. Since Schema supports only single inheritance, subdominant types are "mixed in," preferably in a common and uniform manner.

OAGIS 8 establishes the following design conventions:

- Names appear as unabbreviated CamelCase: elements and types use UpperCamelCase; attributes use lowerCamelCase.
- Content is modeled predominantly in element-normal form for maximum extensibility.
- No anonymous type definition; all types are named.
- Type names are typically the same as their corresponding element names (Attachment of type Attachment).
- Element names don't contain parent element names (Documents have Attachments, not DocumentAttachments – exceptions include the "PersonName" component, since the global element "Name" is too generic).
- Each vocabulary exists in a separate namespace. Only core OAGIS content lives in the OAGIS namespace.

Fields and Compounds, OAGIS's basic building blocks, are defined as types (simple and complex, respectively), and bound as local elements in Components and Nouns. Fields and Compounds are neither UserArea-extensible nor overlay-extensible.

Components are OAGIS's larger-grained building blocks, with complexContent (element children) made up of Fields, Compounds, and Components. Components are UserArea-extensible and overlay-extensible. Each Component is represented by a global type and a global element (uniquely named).

Nouns are overlay-extensible and are represented by both global type and global element. Base Nouns are a factorization of shared Noun content. For two or more Nouns that share common, dominant content, the shared content is abstracted into a Base Noun. Examples include Document, Order, and Ledger. The following code shows a PurchaseOrder Noun, an extension of Base Noun Order:

```
<xs:element name="PurchaseOrder" type="PurchaseOrder"
    substitutionGroup="Noun"/>
<xs:complexType name="PurchaseOrder">
 <xs:complexContent>
  <xs:extension base="Order"/>
 </xs:complexContent>
</xs:complexType>
```

AddRequisition	**Get**Pricelist	**Get**ShipmentSchedule
CancelRequisition	**Show**Pricelist	**Show**ShipmentSchedule
ChangeRequisition	**Sync**Pricelist	
GetRequisition		**Load**Receivable
ShowRequisition	**Add**PurchaseOrder	
	CancelPurchaseOrder	**Get**Invoice
AddRequestForQuote	**Change**PurchaseOrder	**Process**Invoice
CancelRequestForQuote	**Get**PurchaseOrder	**Show**Invoice
ChangeRequestForQuote	**GetList**PurchaseOrder	**Sync**Invoice
GetRequestForQuote	**Process**PurchaseOrder	
RespondRequestForQuote	**Receive**PurchaseOrder	**Create**MaintenanceOrder
	ShowPurchaseOrder	**Cancel**MaintenanceOrder
AddQuote	**Sync**PurchaseOrder	**Get**MaintenanceOrder
CancelQuote		**Cancel**MaintenanceOrder
ChangeQuote	**Create**ProductionOrder	
GetQuote	**Cancel**ProductionOrder	**Get**InventoryCount
ShowQuote	**Get**ProductionOrder	**Show**InventoryCount
SyncQuote	**Show**ProductionOrder	**Sync**InventoryCount
	SyncProductionOrder	

TABLE 1 | Some characteristic OAGIS 8 BODs

Further factorization can occur, resulting in, for example, a "FinancialDocument" Base Noun.

Following Schema rules, all extended content appears after base content. By convention a UserArea, when allowed, is the last child element.

Representation Challenges: How They Were Addressed

At the conclusion of any ambitious project, it's useful to reflect on what went well and what could have gone better. Looking at OAGIS's move to Schema, several Schema capabilities are clear winners:

- A vastly improved type system, with named simple and complex types
- Support for XML Namespaces
- Substitution group extensibility
- Local and global element definitions
- Assembly of modular schema files
- Widespread support by other XML technologies, tools

Building on these strengths we were able to achieve all of our major OAGIS 8 design goals. There were a number of challenges, though, involving a few key Schema features:

- Complex type derivation by restriction
- Inextensible enumerations
- Poor/incomplete metaobject representation
- The tease of almost having Object semantics (but, hey – it's markup, not a universal representation language)
- The forced flattening of namespace to get OO-like behavior

The issues encountered have been shared with the W3C Schema Working Group; we hope to see these concerns addressed as the Schema Rec evolves. Details of the major challenges – and how they were addressed – are laid out below.

Derivation by restriction of complex types

The Schema feature that was least compatible with OAGIS 8 design goals was derivation by restriction of complex types. It affected OAGIS in two places: in implementing Overlay Extensions (addressed above) and in attempting to support multiple cardinality models for the "the same" Noun.

Schema extensibility, as found in introductory texts, is relatively simple: take a type, derive a new type by extension, and then use that new type. It's that simple. But what if you have a component (e.g., OrderStatus) that's widely used (in PurchaseOrder, SalesOrder,…) and you'd like to extend it a little? Simple: find its original type definition, add some content, and go. But what if you can't touch that component's type because it's a frozen part of a stable specification? Without resorting to substitution groups, Schema provides one alternative – derivation by restriction – which, for each thing that uses the new component, forces replication of all inherited content. When you replicate content, you must maintain each replicant. We all know that replication is bad, yet it's an inherent part of derivation by restriction.

A related problem arises when extending: extension requires restriction, and extension and restriction must occur in separate derivation steps. To extend a thing, you must create an extra intermediate type – one for the restriction step and another for the extension step. Consider keeping track of several content model replicates plus their accompanying intermediate derivation steps. The situation quickly grows unmanageable for any practical application.

On top of everything else, there's one key aspect of derivation by restriction that renders it incompatible with OAGIS 8: there's no practical way to derive by restriction across namespace boundaries. Since OAGIS overlays live in separate namespaces, derivation by restriction is incompatible with overlay extensibility.

"BOD constraints" used instead of derivation by restriction

A key manifestation of the issues with derivation by restriction was Noun Normalization: maintaining consistency of multiple uses of the same Noun with different content cardinalities. How do you establish different sets of required and optional parts of a thing, given how/where the thing is used? That's the challenge we faced with having different treatments of each Noun, depending on which Verb is applied to it.

For example, Canceling a PurchaseOrder typically requires only a subset of PurchaseOrder information to identify which PurchaseOrders to cancel (any combination of content could be used), but Processing a PurchaseOrder requires the presence of a great deal of the content. How does one specify that any content can be omitted for a CancelPurchaseOrder BOD, yet little content can be omitted from a ProcessPurchaseOrder? The options boiled down to essentially two: replicate the Noun content for each use of the Noun (and maintain the separate replicants), or relax the cardinality constraints entirely and rely on something other than Schema validation to enforce the constraints. No matter what combination of Schema features was attempted (and virtually all were), it always came to that.

Given how complex and error-prone replication can be (especially in the face of overlay extensibility), OAGIS architects opted for the latter – to relax all minimum occurrence constraints within each Noun and to establish separate BOD constraints, represented as XPath expressions, for example, in Schematron rules processible as XSLT. OAGIS defines a set of rules per BOD, expressing the true minimum occurrence constraints for all nonoptional Noun parts, plus any other useful constraints (e.g., famed "co-occurrence" constraints). There are few constraints on the CancelPurchaseOrder BOD and quite a few on ProcessPurchaseOrder.

BOD constraints are typically evaluated after schema validation has screened for correct structure and type. Since most XSLT processors can use schema-validating parsers, both types of validation could be applied in one parse. Preferably, Schema could incorporate at least rudimentary constraints, and schema-validating parsers could validate those constraints.

Inextensible enumerations

Whereas DTDs were hyperextensible, Schema extensibility is much more controlled. Enumerations in Schema are based on derivation by restriction of simpleTypes – for example, by restricting the set of all strings to the set containing all known currency codes. There's no notion of extending a Schema simpleType, only restricting. Need to add a new currency code? It can't be done without perturbing the fixed, stable specification.

With enumerations in Schema inextensible, a significant capability became impossible in OAGIS 8. Fields such as ChargeType, with values BasicFreight, OriginPort, and Miscellaneous, couldn't be both extended and validated. Being inextensible, only truly stable valuesets qualify to be enumerations: days of the week, yes-no-answer, and the like. Any enumeration with the slightest volatility (e.g., CurrencyCode) must be handled in another fashion: either nonvalidated or application-validated. A practical, extensible alternative to Schema's enumerations is clearly needed.

Using "semantically named elements" to address inextensible enumerations

For semistable items that cry out for Schema validation, OAGIS 8 used an alternative mechanism we called *semantically named elements*. This approach uses substitution groups and first-class elements instead of a single element with a pseudo-type-name field. Rather than have a Charge element containing a ChargeType with various enumerated values, the enumerations are promoted to element names, replacing the catchall Charge element with the more specific BasicFreightCharge, OriginPortCharge, MiscellaneousCharge elements, all of type Charge (or some derivation).

This is goodness in a specification that originated in the era of relatively flat, catchall tables and endeavors to be more object oriented. It even supports type differentiation where warranted; for example, a ShipToParty can have loading dock hours properties, which are irrelevant to a BillToParty. However, it's a bit more of a challenge for table-oriented applications, which must map those nicely differentiated objects back into tables.

Substitution is for globals only

To have true, plug-in extensibility – the only kind that works for overlays – all extensible elements must be global (Schema doesn't support substitution for local elements). Since global elements must be globally unique, the names of extensible components end up being longer than we'd like – an Order's Status element becomes the global OrderStatus element. Globally unique names are so…DTD-ish.

Using "local-globals" when global names are too specific

Let's say you want BODs to have extensible "Header" and "Line" child elements. Since overlay-extensible components are global elements they must be uniquely named, yet the names Header and Line are too generic to be global and unique. Rather than force arcane and nonuniform naming conventions, a workaround is to use a file-based binding of "local globals," based on the assumption that there will only ever be one Header or Line in a given BOD. Better would be a Schema-supported means of substituting for local elements.

Metamodeling deficiencies

All great models spring from profound metamodels. And having a representation that establishes metalevel objects (BusinessObject-Document, Verb, Noun,…) and imposes constraints on the models ("all BusinessObjectDocuments have DataArea with a Verb and a Noun") can be a real benefit. Try as we might, we just couldn't find a workable representation in Schema for many of our metalevel objects.

Namespaces and extensions – gotta map?

With all of OAGIS's representation challenges addressed, other issues remain. Most vexing is being object structured in an as-yet tabular world. Many XML middleware solutions relied on relatively easy-to-map, nonnamespace-differentiated DTD representations, and are reeling at the challenge of mapping Schema's more deeply structured, namespace-diverse representations into their table structures. Add in the new Schema extensibility concepts of derivation by extension and substitution groups, and these vendors have their plates full in accommodating Schema. With time and customer demand these issues will diminish, yielding worthwhile gains.

Observations

▸ During OAGIS 8 design, and subsequently in its application by users, we found that developing usable Schemas of any practical complexity requires the use of a schema-savvy IDE.
▸ Beware variability among parsers! Interpretations of the Schema Rec continue to vary, with strongest compliance observed among production parsers. Any schema to be used in production must be checked against all production parsers, to ensure universal compliance.
▸ Simple examples make W3C XML Schema look easily extensible. Real practice shows that true extensibility is more complicated, and all but requires the use of substitution groups. Any other approach requires complex type derivation by restriction, which quickly becomes unmanageable.
▸ Even the best practices are best only if they work in practice. All XML Schema architectures must be validated/verified with numerous test cases. That's because the Schema Rec isn't easy to read and correctly apply. An error-free parse of the schema alone can lull one into a false confidence. It's good practice to check all schemas for internal correctness and to verify that planned examples don't violate provisions of the Rec (unique particle attribution is a real killer).
▸ Be prepared for breakage between parser releases as vendors build out their compliance checking. This gap continues to be narrowed as parser vendors converge on full compliance.

• • •

OAGIS architects have achieved far more with Schema than was ever possible with DTDs, both in addressing the long-standing issues of an early XML adopter and in realizing OAGIS's groundbreaking new capabilities. Without Schema, OAGIS 8 would not have been able to strengthen its established, horizontal base, and certainly couldn't have addressed the accelerating need for vertical extensibility. OAGIS 8 will succeed, largely due to XML Schema's strengths.

The OAGIS 8 project addresses an extremely complex XML model, certainly pushing the boundaries of Schema. Complex, creative solutions – especially those that appear simple to users of the model while hiding significant complexity faced by the developers of the model – were often called for. To ensure Schema's future success, the Schema Working Group must continue to attend to the practical necessities of Schema use. Schema is a huge step forward, but it can and should continue to improve. That's the natural evolution of an effective standard.

OAGIS 8, the Schema-based, vertically extensible integration specification, is freely available to everyone at www.openapplications.org.

Acknowledgments

I'd like to acknowledge the other members of the OAGIS 8 architecture team: Michael Rowell, chief architect, OAGI; Kurt Kanaskie, Lucent Technologies; Andrew Warren, Canopy International; Satish Ramanathan, MRO Software; with much help from Duane Krahn, Irista.

AUTHOR BIO

Mark Feblowitz, XML architect for Frictionless Commerce, is on loan to the Open Applications Group, where he played a leadership role in adapting OAGIS to XML Schema. Mark is also team lead for OAG contracts, developing BODs for contract interchange. Mark's XML representation background derives from years of applied research in conceptual modeling and knowledge representation and reasoning. Previously, Mark spent nearly 20 years at GTE Laboratories (now Verizon Laboratories) in enterprise requirements modeling.

MFEBLOWITZ@FRICTIONLESS.COM

LISTING 1 Populated UserArea in Charge component

```
<Charge>
 <Id>String</Id>
 <Total currency="String">3.14</Total>
 <Cost>
  <Amount currency="String">3.14</Amount>
  <FunctionalAmount currency="String">3.14</FunctionalAmout>
  <PerQuantity uom="Each">3.14</PerQuantity>
 </Cost>
 <Description lang="en-us"
 owner="String">String</Description>
 <Distribution/>
<UserArea xmlns:myns="MyNamespace"
      xsi:schemaLocation="MyNamespace
      http://www.MyCo.com/MyStuff.xsd">
  <EffectivePeriod>
   <From>2002-04-05</From>
   <Duration>P1Y2M3D</Duration>
  </EffectivePeriod>
  <myns:QualityScore>100.0</myns:QualityScore>
 </UserArea>
</Charge>
```

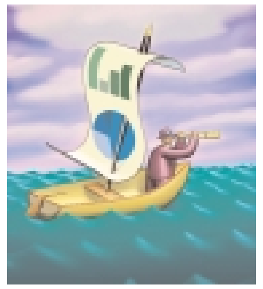
Download the Code
www.sys-con.com/xml

Chart Your Course to Success in IT...

...order your copy of Java Trends: 2003

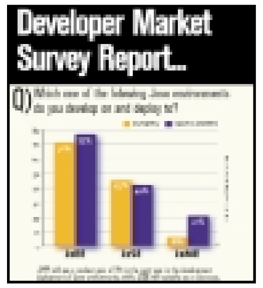
Don't go astray. In the vast sea of Internet technology, market conditions change constantly. Will Java remain the hot platform it is today? Will C# rapidly gain ground? What are the world's foremost Java developers aiming their sights toward? Which companies come to mind when planning their next project? How can their thinking direct your efforts?

EnginData's research is invaluable to leading IT vendors, integrators, Fortune 500 companies, trade-show organizers, publishers, and e-business organizations worldwide.



EnginData Research has studied the IT industry extensively, spotting many key trends and alerting industry leaders along the way. In the first quarter of 2002, they embarked on their most challenging mission ever: the most in-depth study ever done of the Java development marketplace.

After months of analyzing and cross-referencing more than 10,500 comprehensive questionnaires, the results are now available.



Here's just a sample of the critical data points the Java report delivers...

- ✔ Current and projected usage of languages and platforms
- ✔ Multiple rankings of hundreds of software vendors and tools
- ✔ Types of applications being developed
- ✔ Databases being used
- ✔ Purchasing patterns
- ✔ Company size
- ✔ Development and purchasing timelines
- ✔ Perceptions and usage of Web services
- ✔ Preferred Java IDE
- ✔ J2EE, J2ME, J2SE usage comparisons

As an IT specialist, **EnginData Research** focuses exclusively on three leading drivers in IT today – Java, Web services, and wireless development. Coming soon, our Web services survey will deliver such critical knowledge as:

- ✔ Time frame for developing Web services – enabled apps
- ✔ Percentage of apps with Web services today
- ✔ Sourcing and hosting Web services
- ✔ Perceived leaders in Web services tools and solutions

Navigate your company through the challenging IT marketplace with the trusted knowledge and intelligence of **EnginData Research**. Order your copy of the 2002–2003 Java market study by calling Margie Downs at 201-802-3082, or visiting our Web site.

engindata RESEARCH

www.engindata.com

VoiceXML CCXML SALT

Architectural tools for enabling speech applications

Written by Ian Moraes

There's been an industry shift from using proprietary approaches for developing speech-enabled applications to using strategies and architectures based on industry standards. The latter offer developers of speech software a number of advantages, such as application portability and the ability to leverage existing Web infrastructure, promote speech vendor interoperability, increase developer productivity (knowledge of speech vendor's low-level API and resource management is not required), and easily accommodate, for example, multimodal applications. Multimodal applications can overcome some of the limitations of a single mode application (GUI or voice), thereby enhancing a user's experience by allowing the user to interact using multiple modes (speech, pen, keyboard, etc.) in a session, depending on the user's context.

VoiceXML, Call Control eXtensible Markup Language (CCXML), and Speech Application Language Tags (SALT) are emerging XML specifications from standards bodies and industry consortia that are directed at supporting telephony and speech-enabled applications. The purpose of this article is to present an overview of VoiceXML, CCXML, and SALT and their architectural roles in developing telephony as well as speech-enabled and multimodal applications.

Before I discuss VoiceXML, CCXML, and SALT in detail, let's consider a possible architectural deployment that employs these specifications. At a high level are two main architectural components: document server and speech/telephony platform. Each interfaces with a number of secondary servers (Automated Speech Recognition server (ASR), Text-to-Speech server (TTS), data stores).

In this architecture a document server generates the documents in response to requests from the speech/telephony platform. The document server leverages a Web application infrastructure to interface with back-end data stores (message stores, user profile databases, content servers) to generate VoiceXML, CCXML, and SALT documents. Typically, the overall Web application infrastructure separates the core service logic (the business logic) from the presentation details (VoiceXML, CCXML, SALT, HTML, WML) to provide a more extensible application architecture. The application infrastructure is also responsible for maintaining application dialog state in a form that's separate from a particular presentation language mechanism.

To process incoming calls, the speech/telephony platform requests documents from the document server using HTTP. A VoiceXML or CCXML browser that resides on the platform interprets the VoiceXML and CCXML documents to interact with users on a phone. Typically, the platform interfaces with the PSTN (Public Switched Telephone Network) and media servers (ASR, TTS) and provides VoIP (SIP, H.323) support. An ASR server accepts speech input from the user, uses a grammar to recognize words from the user's speech, and generates a textual equivalent that is used by the platform to decide the next action to take, depending on the script. A TTS server accepts markup text and generates synthesized speech for presentation to a user. In this deployment a SALT browser on a mobile device interprets SALT documents. Figure 1 is a diagram illustrating such an architecture.

VoiceXML

Now that you have an overall understanding of the architecture in which these specifications can be used, let's begin by discussing VoiceXML. VoiceXML can be viewed as another presentation language (HTML, WML) in your architecture. VoiceXML is a dialog-based XML language that leverages the Web development paradigm for developing interactive voice applications for devices such as phones and cell phones. It's a self-contained presentation language designed to accept user input in the form of DTMF (touch tones produced by a phone) and speech, and to generate user output in the form of synthesized speech and prerecorded audio. It isn't designed to be embedded in an existing Web language (e.g., HTML, WML) and leverage HTML's event mechanism. At this writing, VoiceXML 2.0 is a W3C working draft (www.w3.org/TR/voicexml20/).

VoiceXML is currently used to support a number of different types of solutions – for example, to automate the customer care process for call centers and to support sales force automation to enable a sales agent to access appointments, customer information, and address books by phone. It's also used in unified communications solutions to enable a user to manage his messages (e-mail, voice, fax), including personal information (personal address books).

To appreciate how VoiceXML can be used, it's necessary to understand its structure, elements, and mechanisms. A VoiceXML application comprises a set of documents sharing the same application root document. When any document in the application is loaded, the root document is also loaded. Users are always in a dialog defined in a VoiceXML document (i.e., the interaction between a user and voice/telephony platform is represented in a dialog). Users provide input (DTMF or speech) and then, based on application logic in the document, are shown a new dialog that presents output (audio files or synthesized speech) and accepts further user actions that can result in a new dialog. Execution ends when no further dialog is defined. Transitions between dialogs use URIs.

There are two main types of dialogs: forms and menus. Forms present output and collect input while menus present a user with choices. Fields are the building blocks of forms and comprise prompts, grammars (describing allowable inputs), and event handlers. VoiceXML's built-in Form Interpretation Algorithm (FIA) determines the control flow of a form. For each form element the main FIA routine involves selecting a form item, collecting input (playing a prompt, activating grammars, waiting for input or event), and then processing the input or event. Examples of actions defined in a form include collecting user input (<field>), playing a prompt (<prompt>), or executing an action when an input variable is filled (<filled>). The FIA's interpretation of a form ends when an <exit> element or transition to another dialog (<submit>, <goto>) is encountered.

Menus (<menu>) can be loosely described as a special case of a form with a single, implicitly defined field composed of a set of choices (<choice>). A choice element (<choice>) defines a grammar that determines its selection and a URI to transition to. Menus can be speech, DTMF, or both.

VoiceXML supports both system-directed (for less experienced users) and mixed-initiative conversations (for more experienced users); <link> is used to support mixed-initiative dialogs. VoiceXML defines specific elements to control dialog flow and handle user interface events. Elements such as <var>, <if>, and <goto> are provided to define application logic, and ECMAScript code can be defined in <script> elements. VoiceXML offers elements to handle situations when there's no user input or the input isn't understandable through its event mechanism (<throw>, <catch>, <noinput>, <error>, <help>, <nomatch>). These and other VoiceXML elements are used in Listing 1 to illustrate how VoiceXML could be used to support a simple-content services application (voice portal for getting stock quotes, news, weather, etc.).

VoiceXML's grammar activation is based on the scope in which the grammar was declared and the current scope of the VoiceXML interpreter. For example, declaring a grammar in the root document means that the grammar will be active throughout the execution of the VoiceXML application. Grammars can be active within a particular document, form, field, or menu, and can be inline, external, or built in (e.g., Boolean, date, phone, time, currency, digits). Although, unlike VoiceXML 1.0, it doesn't preclude other grammar formats, version 2.0 requires support for the XML form of the Speech Recognition Grammar Specification (SRGS). VoiceXML 2.0 interpreters should also be able to support Speech Synthesis Markup Language (SSML) specification for synthesized speech. As I write, both SSML and SRGS are W3C working drafts.

VoiceXML has some other features to promote architectural extensibility and reusability. For example, the <object> element offers facilities for leveraging platform-specific functionality while still maintaining interoperability, and the <subdialog> element can be used to develop reusable speech components.

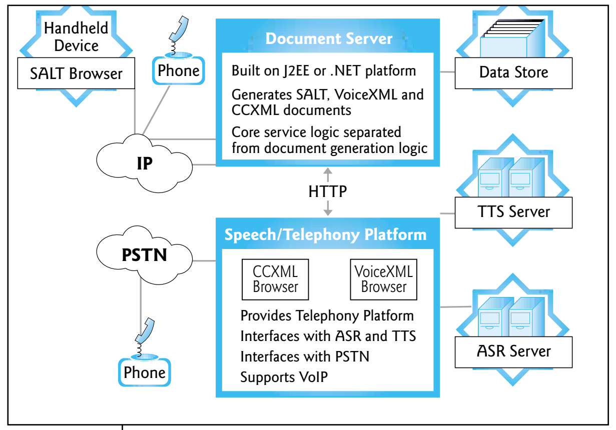

Handheld Device
SALT Browser
Phone
Document Server
Built on J2EE or .NET platform
Generates SALT, VoiceXML and CCXML documents
Core service logic separated from document generation logic
Data Store
IP
HTTP
TTS Server
PSTN
Speech/Telephony Platform
CCXML Browser
VoiceXML Browser
Provides Telephony Platform
Interfaces with ASR and TTS
Interfaces with PSTN
Supports VoIP
ASR Server
Phone

FIGURE 1 Architectural illustration of possible VoiceXML, CCXML, and SALT deployment

While it provides a suitable language for developing interactive speech applications, VoiceXML lacks support for features such as conferencing, call control, and the ability to route and accept or reject incoming calls. <transfer>, its tag for transferring calls, is inadequate for these types of features. Further, VoiceXML's execution model isn't well suited to an environment that needs to handle asynchronous events external to the VoiceXML application. VoiceXML can handle synchronous events only – those that occur only when the application is in a certain state.

CCXML

CCXML addresses some of the telephony/call control limitations of VoiceXML. It enables processing of asynchronous events (events generated from outside the user interface), filtering and routing of incoming calls, and placing outbound calls. It supports multiparty conferencing as well as the creation/termination of currently executing VoiceXML instances and the creation of a VoiceXML instance for each call leg. A CCXML browser on a voice/telephony platform interprets CCXML documents. CCXML is currently a W3C working draft (www.w3.org/TR/ccml/).

Since CCXML is still an emerging specification, few deployed solutions are in the market today. However, it can be used to support a number of different types of applications. Conferencing applications can be developed using it. Voice messaging applications can use it to enable a user to filter incoming calls and route them to a particular application. CCXML can also be used to support notification functions (e.g., place outbound calls to notify a user of a new appointment or a stock alert).

The structure of a CCXML program reflects its fundamental value in being able to handle asynchronous events. Processing of events from external components, VoiceXML instances, and other CCXML instances is central to CCXML. A CCXML program basically consists of a set of event handlers (<eventhandler>) for processing events in the CCXML event queue. Each <eventhandler> element comprises <transition> elements. An implicit Event Handler Interpretation Algorithm (EHIA) interprets <eventhandler> elements. A CCXML interpreter essentially removes an event from its event queue, processes it (selects <transition> elements in <eventhandler> that match the event and performs actions within that <transition> element), and then removes the next item from the event queue.

Within a <transition> element a new VoiceXML dialog can be started (<dialogstart>) and associated with a call leg. Note that the dialog launch is nonblocking, and control is immediately returned to the CCXML script. A CCXML script can also end a VoiceXML dialog instance (<dialogterminate>). Conditional logic (<if>, <else>) can be used in a <transition> element to accept or reject incoming calls or modify control flow. Incoming calls can be accepted or rejected (<accept>, <reject>), and outbound calls can be placed (<createcall>). A CCXML application can contain multiple documents, and a CCXML execution flow can transition from one document to another (<goto>, <fetch>, <submit>) and end its execution (<exit>). A CCXML instance can also create another CCXML instance (<createccxml>) with a separate execution context and send events (<send>) to other CCXML instances. Multiparty conferences can be created and terminated (<createconference>, <destroyconference>), and call legs can be added to and removed from a conference (<join>, <unjoin>).

For telephony events the JavaSoft call model is used to provide the abstractions (Address, Call, Connection, Provider). A Call object is a representation of a telephone call. A call comprises zero or more Connections (e.g., a conference call typically has three or more connections). A Connection describes the relationship between a Call and an Address (e.g., telephone number). A Connection is in one of a defined set of states at a particular time. For example, when a Connection and an Address are an active part of a call, a specific event is emitted (connection.Connection_CONNECTED), whereas when an address is being notified of an incoming call, a different event is emitted (connection.CONNECTION_ALERTING).

The code snippet in Listing 2 illustrates the main CCXML elements involved in processing an incoming call notification event and then processing a subsequent connected event. Based on a connection. CONNECTION_ALERTING event, the call could be accepted or rejected according to the called ID using conditional logic (<if>, <else>). When a connection enters the connected state, a VoiceXML dialog (login.vxml) is launched to perform some authentication (e.g., entering a PIN number). Listing 2 also illustrates the main structure of a CCXML script to address this scenario.

Other types of events processed by a CCXML script are sent from VoiceXML instances or from other CCXML instances. For example, a CCXML script can capture status from a terminating VoiceXML dialog because VoiceXML's <exit> element allows variables to be returned to the CCXML script. When a VoiceXML interpreter ends execution, dialog.exit event is received by a CCXML instance. Using a <transition> element, the CCXML script can process the dialog.exit event and access variables returned by the VoiceXML dialog. For example, if we have a VoiceXML dialog that presents a set of menu options, upon termination of the VoiceXML dialog the CCXML script can obtain the selected menu choice and act on it (e.g., perform an outdial, join a conference). Events are also used to communicate between CCXML instances. However, CXXML currently doesn't define a specific transport protocol for communication; SIP and HTTP are possibilities for the underlying transport.

A number of factors currently inhibit the adoption of CCXML.

- Since it's a new specification still under review, there are few CCXML browser implementations compared to VoiceXML browser implementations. Note that although it's designed to complement a dialog-based language such as VoiceXML, a CCXML system isn't required to support a VoiceXML implementation. If CCXML and VoiceXML are used together, the instances would run separately.
- While CCXML is a promising start in addressing some of the limitations of VoiceXML, a number of areas remain to be specified in order to meet the needs of call control applications. For example, while CCXML is and should remain uncoupled from a specific underlying protocol, protocol-agnostic mechanisms could be specified to allow passing in protocol-specific parameters (VoIP, SS7) when performing certain functions (e.g., making an outbound call).
- Other items for future discussion are documented in the current CCXML specification (e.g., communication between different CCXML instances).

SALT

SALT enables speech interfaces to be added to existing presentation languages (e.g., HTML, XHTML, WML) and supports multimodal applications. PDAs, PCs, and phones are examples of devices that can support SALT applications. At the time of this writing, the SALT Forum (www.saltforum.org), an industry consortium, has published a 0.9 version of the SALT specification.

" These XML specifications are posing an exciting challenge for developers to create useful, usable, and portable speech-enabled applications that leverage the ubiquitous Web infrastructure "

Since SALT is an emerging specification, at the time of this writing there are no known deployed solutions. However, SALT can enhance a user's experience on PDAs and other mobile devices that have inherent limitations, such as keyboards that are difficult to use and small visual displays. Multimodal solutions allow a user to use multiple I/O modes concurrently in a single session. Input can be via speech, keyboard, pen, or mouse, and output can be via speech, graphical display, or text. For example, a multimodal application can enable a user to enter input via speech and receive output in the form of a graphic image. Thus a user can select the appropriate interaction mode depending on the type of activity and context. SALT can enable solutions that enhance a user's experience when using applications such as content services (news, stock quotes, weather, horoscope), unified communications, sales force automation, and call center support.

The core philosophy behind the SALT specification is to use a lightweight approach to speech-enabling applications. This is evident in the small set of tags in the SALT specification that can be embedded in a markup language such as HTML for playing and recording audio, specifying speech synthesis configuration, and recognizing speech. The <listen> element is used to recognize input and define grammars (<grammar>). SALT browsers must support the XML form of the W3C's SRGS.

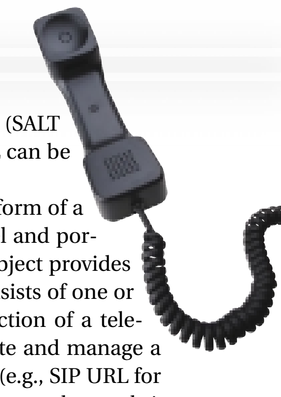
This means that the XML form of SRGS can be used in SALT's <grammar> element. The <listen> element also has an element for processing the input (<bind>). The <bind> element processes the input result that's in the form of a semantic markup language document. A SALT browser must support W3C's Natural Language Semantic Markup Language (NLSML) for specifying recognition results. XPath is used to reference specific values in the returned NLSML document that are then assigned to a target defined in the <bind>. Thus input text can be obtained from either a graphical display or, now, using a speech interface.

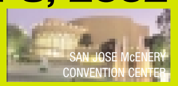
The <listen> element can also contain a single <record> element for capturing audio input. SALT also supports DTMF input using the <dtmf> element; like <listen>, its main elements are <grammar> and <bind>. For audio output the <prompt> element can represent a prompt using a speech synthesis language format (SALT browsers must support SSML). This means that SSML can be embedded within SALT's <prompt> elements.

SALT's call control functionality is provided in the form of a call control object model based on the JCP Call Model and portions of Java Telephony API (JTAPI). The callControl object provides an entry point to the call control abstractions and consists of one or more provider objects. Providers represent an abstraction of a telephony protocol stack (SIP, H.323, SS7). Providers create and manage a conference. An address is a communication endpoint (e.g., SIP URL for VoIP), and providers define the available addresses that can be used. A conference is composed of one or more calls that can share media streams. SALT's call control objects define states, transitions, events, and methods for supporting call control. For example, there are methods for beginning a new thread of execution for processing an incoming call (spawn() method of the callControl object), answering an incoming call (accept() method of call object) based on an alerting event (call.alerting), or transferring a call (transfer() method of call object).

Also, SALT has a <smex> element that can be used to exchange messages with an external component of the SALT platform. Functionality such as Call Detail Record (CDR) generation, and logging or proprietary features, can be leveraged using this extensible mechanism while maintaining interoperability. For example, <smex> can be used to leverage call control functionality on a different platform (e.g., send call control requests to and receive events from a CCXML interpreter on a remote host). In this situation SALT can be used for dialog management while call control is handled by a separate architectural entity (e.g., CCXML interpreter) on a remote host.

Let's consider how SALT elements can be embedded in an existing Web page to also offer a speech interface. Assume a Web page with SALT elements allows a user to select the latest news for a particular sport (soccer, basketball, football, tennis) by selecting the sport from a drop-down box. To enable speaking the name of the sport, the start() method of listen's DOM object is called to invoke the recognition process. The value returned





from the recognition is assigned to an input field (e.g., txtBoxSport). As with other SALT DOM elements, the listen element also defines a number of events (e.g., onreco, onsilence, onspeechdetected) whose handlers may be specified as attributes of the listen object (i.e., event-driven activation is an inherent feature of SALT). The following code snippet illustrates some of the SALT elements involved in this scenario:

```
…
<input name="txtBoxSport" type="text" onpendown=
 "listenSport.start()"/>

<salt:listen id="listenSport">
<salt:grammar name="gramSport" src="/sport.xml"/>
<salt:bind targetElement="txtBoxSport" value="//sport"/>
</salt:listen>
…
```

You've probably noted that SALT and VoiceXML can be used to develop dialog-based speech applications, but the two specifications have significant differences in how they deliver speech interfaces. Whereas VoiceXML has a built-in control flow algorithm, SALT doesn't. Further, SALT defines a smaller set of elements compared to VoiceXML. While developing and maintaining speech applications in two languages may be feasible, it's preferable for the industry to work toward a single language for developing speech-enabled interfaces as well as multimodal applications.

Summary and Conclusion

This short discussion has provided a brief introduction to VoiceXML, CCXML, and SALT for supporting speech-enabled interactive applications, call control, and multimodal applications and their important role in developing flexible and extensible standards-compliant architectures. This presentation of their main capabilities and limitations should help you determine the types of applications for which they could be used.

The various languages expose speech application technology to a broader range of developers and foster more rapid development because they allow for the creation of applications without the need for expertise in a specific speech/telephony platform or media server. The three XML specifications offer application developers document portability in the sense that a VoiceXML, CCXML, or SALT document can be run on a different platform as long as the platform supports a compliant browser.

These XML specifications are posing an exciting challenge for developers to create useful, usable, and portable speech-enabled applications that leverage the ubiquitous Web infrastructure.

References

For more information on using these XML specifications in your speech-based system architectures, see the following:

- *Voice browser call control (CCXML v. 1.0):* www.w3.org/TR/ccxml
- *Speech Application Language Tags 0.9 Specification (draft):* www.saltforum.org
- *Voice Extensible Markup Language (VoiceXML v. 2.0):* www.w3.org/TR/voicexml20
- *Speech Synthesis Markup Language Specification:* www.w3.org/TR/speech-synthesis
- *Speech Recognition Grammar Specification for W3C speech interface framework:* www.w3.org/TR/speech-grammar
- *Natural Language Semantics Markup Language for speech interface framework:* www.w3.org/TR/nl-spec

IMORAES@YAHOO.COM

ABOUT THE AUTHOR

Ian Moraes, PhD, develops technical strategies and system architectures for a leading telecommunications solutions provider. He has architected, designed, and developed systems using industry standards such as VoiceXML, J2EE, 3GPP, IETF, and W3C.

LISTING 1 Code snippet of VoiceXML elements for supporting sample content services menu

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

<vxml version="2.0">

   <catch event="event.exit">
     <exit />
   </catch>

   <menu id="choose_menu_option">
      <prompt>
       You are in a demo for content services.
       To obtain stock quotes, say
       stocks or say 1. For weather information,
       say weather or press 2.
       For news, say news or press 3.
      </prompt>

      <choice dtmf="1" next=
       "#stocks_menu_option">stocks</choice>
      <choice dtmf="2" next="#weather_menu_option">
       weather</choice>
      <choice dtmf="3" next=
       "#news_menu_option">news</choice>

      <nomatch>
         <audio>I could not understand what you said,
          please try again</audio>
         <reprompt/>
      </nomatch>

      <noinput>
         <reprompt/>
      </noinput>

      <help>
         <audio>
         …
         </audio>
      </help>
   </menu>

   <form id="stocks_menu_option">
      <block>
      …
      </block>
   </form>

   <form id="weather_menu_option">
      <block>
      …
      </block>
   </form>

   <form id="news_menu_option">
     <block>
     …
     </block>
   </form>
</vxml>
```

LISTING 2 CCXML elements involved in processing incoming call notification event and subsequent connected event

```
<?xml version="1.0" encoding="UTF-8"?>
<ccxml version="1.0">
 …
 <eventhandler statevariable="application_state">

  <transition event="connection.CONNECTION_ALERTING" name="evt">

       …
       <accept/>
  </transition>

  <transition event="connection.CONNECTION_CONNECTED" name="evt">

       …
       <dialogstart src="'login.vxml'"/>
  </transition>

</eventhandler>

 </ccxml>
```

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

John Walker, an independent software developer in Oakland, CA, has over 19 years of experience in application development and relational databases.

REVIEWED BY **JOHN WALKER**

PopChart Server 4.0

by Corda Technologies

Corda Technologies, Inc.
350 South 400 West, Suite 100
Lindon, UT 84042
Phone: 801 805-9500
E-Mail: sales@corda.com
Web: www.corda.com

Test Environment
OS: Dell Precision 340 Workstation running Windows 2000 w/ Service Pack 2
Processor: 1.8 GHz, Intel Pentium IV, 40GB disk
Memory: 512MB

Specifications
Platforms: JRE 1.2 or higher
Pricing: PopChart Server Enterprise Edition, $6,990 including 1 year maintenance; PopChart Builder, $495 including 1 year maintenance

Developers, and users, generally agree that the easiest medium through which to digest data is a graphical one. This rule is particularly true now, considering the volume of data that can be more easily obtained from various distributed data sources using either SQL or XML as a conduit.

Requesting and presenting data in a graphical format, however, has traditionally come at the expense of performance and complexity. Besides the cost of marshaling the data from the disparate sources in order to convert it into a graphical format, image files take a great deal of time to transport over the network and display in client browsers. The number of data points to be represented in a graphical format would normally be directly proportional to the number of performance bottlenecks the system will experience. Introduce the availability of the data to be represented in real time, and that is interactive as well, and most developers would assume that the gains provided by the accessibility of the data in a graphical format would be lost ultimately to the poor application throughput.

Corda Technologies provides both the tools to reduce the complexity when constructing graphical charts and the server to deliver them without hindering performance. Corda's PopChart Server 4.0 delivers a variety of interactive charts representing real-time dynamic data using the latest best-of-breed graphical formats. These formats include Flash, SVG, PNG, GIF, PDF, EPS, WBMP, and descriptive text that is compliant with Section 508 of the Federal Acquisition Regulation for visually impaired users.

Written entirely in Java, PopChart Server 4.0 is a completely portable architecture that can interact with data from a variety of environments. Data can be accessed from XML, flat files, or HTML, directly from a relational database or indirectly from an application server. The PopChart Builder application provides for the quick and easy construction of the appearance files that describe how the data is to be formatted. Furthermore, using the PopChart Builder, developers can describe and implement the dynamic dimensions the charts will represent. For example, if a pie chart represents the employees of a company by country, it would make sense to know how the regions in that country represent those employees. Taking it a step further, a manager using such a chart might desire to drill down even more to see how employees are paid by region. All of this functionality, and more, can be easily facilitated with Corda's products.

Installing / Configuring PopChart Builder and Server

Both PopChart Server and Builder installed easily on the Windows 2000 machine used for this review. No special configuration was necessary other than following the installation screen and providing the software keys for both products. Afterwards it was simply a matter of launching them from the start menu. It's important to mention that Corda recommends that up to 40MB of hard drive space be available for the installation of the Enterprise edition of PopChart Server. It's also suggested that an additional 15MB be available for the PopChart Builder installation.

Another precondition for the installation is the availability of a Java VM compatible with JRE 1.3 or higher. Specifically, Builder needs a 1.3.1 VM while the server needs a 1.2.2 or 1.3.1 VM installed. The installation set makes these VMs available if the current environment doesn't support them. The server runs on Windows 98/NT4.0 or higher, Mac OS X, Solaris 8.0 or higher, and most Unix- or Linux-compatible systems using the equivalent of a Pentium II 233 or higher with 64MB RAM dedicated to your VM. Ideally, 128MB of RAM is desired.

A good variety of sample code is provided for implementing charts in JSP, HTML, JavaScript, Java servlets, ASP, and in the latest version, PHP and ColdFusion. You can plug sample data and appearance files into any of these supporting technologies to get a good feel of how the product is to use and how it functions. I also recommend a visit to www.corda.com to access the variety of powerful real-world examples the company provides to showcase the power of their product.

Working with PopChart Server 4.0

For the purpose of this article, my goal was to create a chart – in this case a pie chart – that organized the employees for a particular company by the country their office was located in. I thought it would also be helpful to see how many employees were associated with a particular region within that country. Finally, the chart had to provide the total payroll for a region and the average employee salary.

In spite of these requirements and an impending deadline, it was very easy to create the format for this chart in the form of an appearance file in Builder. No significant problems were encountered when the file was used within a Java servlet and accessed the data from a relational database. For this example an Oracle9*i*AS application server was used against an Oracle8*i* database.

The first step in creating the chart pictured in Figure 1 was starting the Builder and creating a new project from a template. This option provided me with the PopChart Wizard, which I used to create an appearance file from the robust catalog of templates that come with the product. Builder provides four template families, each containing 12 chart types that can be customized by choosing different color themes and by incorporating dynamic functionality. The chart types include area, bar, line, pie, time line, candlestick stock, and bubble graphs, to name a few.

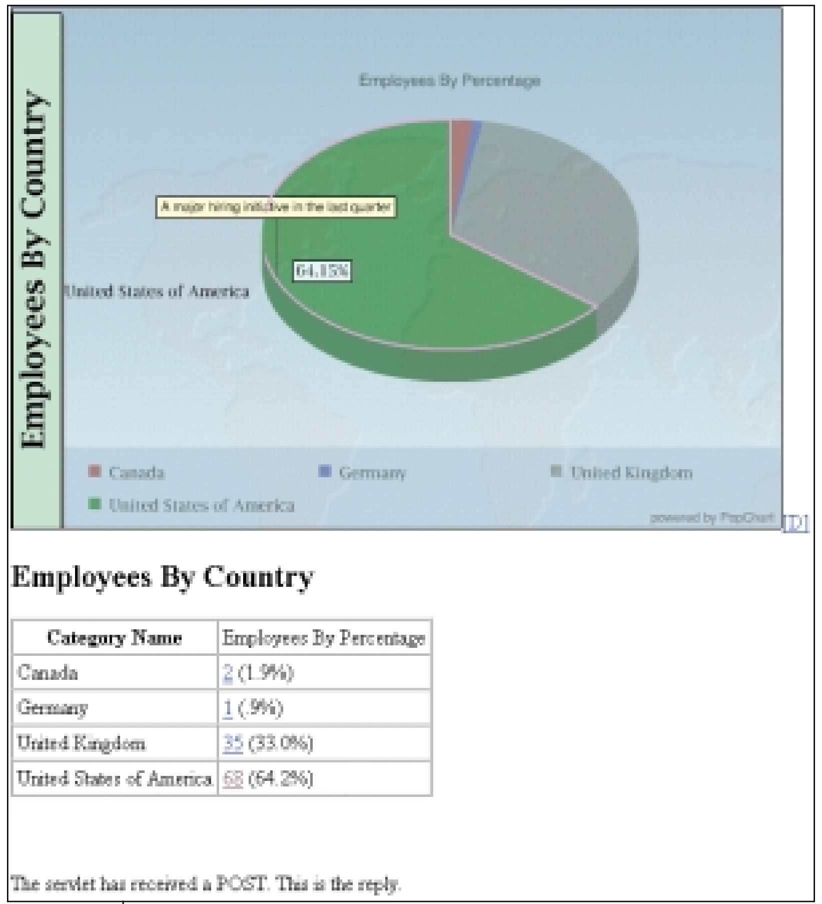
For best image quality, Corda recommends the use of the Macromedia Flash or SVG (Scalable Vector Graphics) formats. These images combine the highest quality resolution with the best performance. They also allow for the interactive capabilities of rollover and popup text, and the ability of the user to drill down on a graph segment to reveal another graph containing more detailed information. Flash and SVG are the only formats that allow for this functionality. However, it's estimated that 96% of Internet users already have the appropriate Flash viewer installed in their browsers for use with PopChart Server. Therefore it's safe to assume that the dynamic functionality will indeed be put to use by your users.

However, if the browser to which the image is being delivered doesn't support either Flash or SVG, then PopChart employs its Best Image Fallback Facility. This feature allows the server to discern what the best format to provide should be, based on the capabilities of the client that is being served. This might mean that instead of seeing the image deployed as a Flash or SVG, a GIF or PNG format might be used. A nice feature of the Builder is that it allows the developer to see how the chart will be displayed in any of these formats. The template can be previewed in the browser so the developer can see simultaneously all of the formats that the appearance file can be displayed in for comparison purposes. Sample data is easily configured for this purpose (see Listing 1).

As a final step, I incorporated the appearance file into a Java servlet for use against live data. A wizard interface is provided to create client code for use against static data in JavaScript, Java servlets, JSPs, ASP, and in the latest version, PHP and ColdFusion. This wizard is an excellent tool for getting the feel of how an appearance file can be incorporated into these technologies. It's not too far from the work necessary to incorporate the same appearance file against dynamic data. The PopChart libraries supply a PopChartEmbedder object that contains the method setDBQuery, which marshals all the necessary data from the relational data source to be incorporated into the graph. After that the developer need only provide the address of the PopChart Server, the path to the appearance file to be used, the name of the data source, and the database driver to be used. A call to the getEmbeddingHTML() method renders the HTML document, along with its supporting JavaScript, which can then be written back to the client.

PopChartEmbedder also has methods for annotating the chart, the desired image format, and whether a link to a Section 508–compliant document should be provided. Section 508 of the Rehabilitation Act of 1973 mandates that federal agencies' electronic and information technology is accessible to people with disabilities. This link appears in the form of a capital "D" in the lower right-hand portion of the screen. When the link is clicked, a document is served describing all of the data in text that, for example, could be served by means of a voice synthesizer to users who are visually impaired.

WALK@SBCGLOBAL.NET



Employees By Country

Category Name	Employees By Percentage
Canada	2 (1.9%)
Germany	1 (.9%)
United Kingdom	35 (33.0%)
United States of America	68 (64.2%)

The servlet has received a POST. This is the reply.

FIGURE 1 | Managing personal financial data

Product Snapshot

Level Beginner to advanced programmers

Pros
- Multiple image format support
- Easy to use Builder tool for creating appearance files
- Powerful API for use with servlets, JSPs, EJBs
- Fast performance

Cons
- Nothing significant

WRITTEN BY **SIG HANDELMAN & PAT SNACK**

XML in the Auto Industry: Summer 2002

Convergence of standards to enable new business collaborations

The automotive industry has been a leader in the use of EDI and EDIFACT over the past 20 years. The deployment of EDI processes down the supply chain has not been uniform, and a common expression to use is the "80/20" rule (e.g., 80% of EDI messages are sent by 20% of the companies). In 1997 the Automotive Industry Action Group (AIAG) did a study, the MAP (Manufacturing Assembly Pilot) project, and estimated a savings of $71 per car if EDI was consistently deployed.

AUTHOR BIOS

Sig Handelman, a former research staff member at IBM TJ Watson Research, has worked on several AIAG work groups, receiving Outstanding Achievement Awards in 2000 and 2002; he is currently cochair of the XML/EDI Work Group. He was technical editor of the Proof of Concept team for the ebXML initiative.

Pat Snack joined the Automotive Industry Action Group in December 2001 on executive loan from General Motors. At AIAG she has oversight responsibility for the Electronic Commerce Steering Committee and its related work groups. At GM, most of her career has been spent in purchasing, logistics, production control, and business information systems.

There has been an everlasting goal of better communication in the supply chain. Complete connectivity would help to minimize risks in the manufacturing process by enabling trading partners to receive timely information on changing situations in manufacturing cycles. The MAP project listed improvements in error rates, reduced lead times, and less inventory as project goals.

When the World Wide Web Consortium released XML, many organizations envisioned that it would replace EDI. The AIAG started the XML/EDI Work Group and saw an opportunity to use XML to achieve the goals of the MAP project. However, there were so many methods of converting EDI to XML that a need for a single standard to use for business was created – ebXML – the organization that has on its masthead "enabling a global electronic market."

ebXML

The work on ebXML, a combination of UN/CEFACT and OASIS, was undertaken by an international group of workers over an 18-month period, and the AIAG and its member companies sent workers to take active roles in developing ebXML. The AIAG, which was represented on the original project teams, participated in one of the first demonstrations of ebXML in Tokyo in November 2000 and in the demonstration the following month in San Francisco. AIAG members have continued to work and lead in the continuing efforts of ebXML in both OASIS (Registry and Messaging) and UN/CEFACT (Business Process and Core Components). (Alan Kotok and Sig Handelman described the relationship between ebXML and the automotive industry in December 2001 in *ActionLine*, the publication of the AIAG.)

In the summer of 2001 the AIAG Work Group channeled its efforts into writing "Opportunity Assessments" of the use of XML and "Guidelines" for XML technology. The former include:

- ***Esmart and COBRA***: Two business processes from Ford Motor Company using XML
- ***Critical response:*** A scenario to define a standard method to handle emergency situations in the supply chain
- ***UDDI scenario document:*** A methodology to define the use of UDDI in the industry

The Guidelines include:

- ***Digital signatures:*** The AIAG is outlining both the technical components of digital signatures and the business processes to use them. We've identified two uses of digital signatures as of this writing: (1) their use in ebXML messaging, and (2) the process of signing XML versions of "Quality Documents," the documents used for the automotive business practice of "signing off" on parts sent from one supplier to another
- ***XML overview:*** An introduction to the XML documents of our work group
- ***XML glossary:*** XML and business terms used in our work group

These documents, written by members of the work group, have been jointly edited during monthly meetings. Access is available to its members through "Eroom," an online collaboration tool licensed by the AIAG from Eroom Technologies.

At the same time, XML issues have been published in the "EDI Planner of the AIAG." Work is progressing to understand how the AIAG's XML messaging standard "E-5" could migrate to a base using ebXML.

In June 2002 the AIAG published a statement of direction (see www.aiag.org/whatsnew._html).The main points follow:

- The AIAG recommends the use of XML 1.0, XML Schemas, and XML Stylesheets.
- The AIAG recommends the use of OAGI Business Object Documents as a start for XML processes. The AIAG is simultaneously working toward the convergence of OAGI BODs and Core Components.
- AIAG supports and endorses the adoption of ebXML as a standard for message routing in an automotive supply chain environment.
  —However, it is recognized that ebXML today is largely untested in our industry.
  —Software incorporating the ebXML messaging services is expected to be widely available to the automotive environment by 2003.
  —As a means of supporting immediate requirements for IP message routing, AIAG continues to support the use of E5-2000, a currently available and tested guideline with automotive functionality.
- The AIAG encourages its members to follow and work on the development of OASIS ebXML Collaboration Protocol Agreements, OASIS ebXML Registry, and the methodologies of UN/CEFACT including Business Process and Core Components.

The statement was approved by the AIAG board of directors.

STAR and ebXML

The AIAG concentrates on issues of the supply chain of the automotive industry. A similar organization is working on the dealer networks that connect car dealers to the factories. This organization, STAR – Standards in Automotive Retail – has been supporting ebXML messaging and was part of the Drummond Group's first round of testing of ebXML messaging. STAR and OAGI are jointly working on OAGI BODs to conduct their business, including parts, inventory, and credit (see www.starstandard.org).

AIAG, OAGI Collaborate to Support XML Strategy Position

AIAG has recognized the need to recommend a set of data communication strategies for the automotive industry. Automotive OEMs and Tier 1's have invested in traditional EDI (based on ANSI ASC X12 or EDIFACT standards) to move data between trading partners. When this technology satisfies the need and provides security and end-to-end reliability, why change?

AIAG understands that EDI will continue to be a viable form of business data exchange for years to come. However, two communities aren't satisfied: (1) the lower tiers, for the most part, continue to receive hard-copy documents, faxes, and telephone calls; and (2) at all levels of the supply chain there continue to be new business processes that require data communication. In both situations, where the investment has not been made, it makes sense to invest in new technology that can provide greater flexibility and functionality for fewer investment dollars. Hence, for XML schema development, AIAG has recommended the use of OAGI BODs and ebXML message routing for transporting the data.

Given the background of AIAG's involvement in international standards development, and the fact that OAGI has been recognized by UN/CEFACT as a capable provider of XML-based payloads, it seemed natural for AIAG and OAGI to collaborate in delivering solutions for the automotive industry. In support of this effort, and for starters, AIAG and OAGI have signed a Letter of Direction defining a formal working relationship.

It's become apparent at AIAG that there is a need to align the various internal work group efforts to a common methodology. With OAGIS 8.0 it's expected that business process models, data models, business process specification sheets, and document formats will take on a consistent appearance across work groups, and that schema development will be harmonized. Core components will be leveraged with existing OAGI BODs and will be reused as new documents are developed. If the foundation is built properly and consistently, when the next-generation technology surfaces these building blocks can be moved to the new syntax.

The exciting piece of this collaboration is that not only is there an opportunity to achieve alignment within the three vertical organizations of automotive (retail, supply chain, and aftermarket), but also horizontally across other industries that participate in OAGI. Critical mass could begin to surface that would potentially enable the enterprise to totally integrate communications and trading partner connectivity. Automotive OEMs have communications needs with companies in each of these related industries, and suppliers typically aren't 100% aligned with automotive. So there are significant benefits to a consistent, open architecture that can work across many vertical industries.

In support of this vision, AIAG and OAGI are collaborating to provide education and training opportunities for their work groups. At the time of this writing August workshops were scheduled at AIAG, specifically to educate the work group members in how to work in OAGIS 8.0. There was to be an Introduction to XML and OAGIS 8.0 workshop for subject matter experts from the business side and an Advanced OAGIS 8.0 workshop for people who will install and deploy the technical side. This offering aims to strengthen the AIAG work groups and attract new work group members. We anticipate that all parties should benefit, as learning will take place in a peer setting with expert instructors from OAGI. Future planning includes offering these materials in formal classes conducted by OAGI and AIAG. In addition, an "Introduction to XML and OAGIS 8.0" was scheduled for the day before the start of the Auto-Tech 2002 conference.

OAGI's August quarterly meeting at the Ritz Carlton in Dearborn, Michigan, was to be held in the automotive community to support the collaboration with AIAG. At this meeting AIAG project leads from specific work groups were to describe their projects, identify the problems needing resolution, and outline data communication needs. Formal OAGI work groups are being established to support the AIAG work groups. The benefit of this relationship is that the broad cross-section of solution providers at OAGI will have marketing needs delivered to their doorstep; AIAG will get expert support from solution providers who can incorporate industry needs into succeeding generations of product. The OAGI/AIAG collaboration should ensure the delivery of successful, more responsive solutions to the automotive industry. In addition, the cross-industry membership of the OAGI work groups will tend to support this same concept horizontally across many industries.

AIAG continues to work globally with its sister associations Odette and JAMA/JAPIA. And we are mutually developing global EDIFACT messages. It's hoped that all three organizations can align on an XML strategy as well.

Web Services

There is a groundswell of support for Web services, an important part of the XML picture. The AIAG's Directory Services and XML Work Groups have taken a start in this area by planning to launch a UDDI pilot in the automotive industry. This pilot would be used to gather information on directory services, registries, and businesses under a common UDDI framework. Several companies in the automotive industry are starting to use Web services for internal and external processes. It should be noted that DaimlerChrysler, one of the automakers, has joined the Web Services Interoperability Organization.

• • •

The AIAG looks to use XML for creating standards for business. The automotive industry is global in scope, and thus we work on the global stage of ebXML. Over the next year we'll be looking for practical methods of using ebXML and OAGIS 8.0. At the same time we realize that both ebXML and Web services will be evolving and we will use the UDDI experience to start working with Web services. Over the coming years the continuing convergence of standards will enable new business collaborations to be developed using XML standards.

S.HANDELMAN@ATT.NET

PSNACK@AIAG.ORG

OAGIS: A 'Canonical' Business Language

Providing both vertical and horizontal requirements

Written by Michael Rowell

It's been said that "the great thing about standards is...there are so many to choose from." Each industry vertical today has its own consortium (for some there are several) that define the standards to be used within that particular industry vertical. In large industries these are broken up into consortia for the different areas. While XML was designed so that multiple data formats can be optimized for different data exchange applications, a problem arises when these different vertical groups define their own message for common business documents independent of one another. For example: request for quotes, purchase orders, invoices, to name a few.

It's natural for us humans to identify with those we are most like and to work with them on common problems. However, it's important for us not to lose sight of the fact that we must provide the information needed to conduct business in a terminology that the parties on each side of the transaction understand. This is true whether we're doing business between companies, within or outside our industry vertical, or up and down the supply chain.

Today what we see are point-to-point integrations by companies in each industry when they need to do business with a company in another industry (see Figure 1). By doing this companies create integrations that don't scale at all when communicating with other companies. Integrating companies in different vertical industries makes this picture worse, as the integrators don't often recognize common constructs because they have different names, structures, and/or naming conventions.

A Horizontal Solution

While it's true that some aspects of a vertical industry are unique to that industry, there are common components and business documents that really are the same or similar. What's needed is a common horizontal language that can be used as a basis for these common components, and common business documents that these vertical industries can plug their additional content, constraints, and terminology into so as to pass information from business to business both inside and outside a given industry vertical (see Figure 2). But this isn't enough.

A company that does business totally within one industry vertical is rare today. For example, Ford Motor Company buys tires from Michelin and gasoline from Exxon, a petroleum company. They also buy office supplies from a retail company – say, Staples. Ford should be able to use one purchase order to communicate with all three companies. Granted, the content will be different, but the XML structure should be the same.

While the purchase order for the petroleum industry may contain information or terminology different from the purchase order for office supplies, the structure and the intent are the same: to buy *n* number of *x* product at *y* cost.

If both the petroleum industry and the office supply retail industry used the same basis for their purchase order, Ford could use the same purchase order business document to buy pencils and gasoline from Staples and Exxon.

An End-to-End Solution

While a horizontal language that enables verticals to plug into a horizontal business language is a significant step forward from where we are today, it doesn't complete the picture. This flow of information cannot stop at the edges of the companies involved; the information must flow back into the business applications that drive these companies as well (see Figure 3).

Without this integration to the back-end systems, neither true end-to-end integration nor the true potential return on investment promised by integrating supply chains can be realized.

At the same time, in today's economy companies need the ability to leverage their current investment in both standards and technology to enable the end-to-end flow of information. They must be sure that this flow will work not only with today's technologies, but with new ones as they're made available.

The "End-in-Mind"

In order to provide the "end-in-mind" of an end-to-end solution, these mechanisms must be provided:

- A mechanism that can add an industry vertical's unique information to the horizontal without modifying the horizontal
- A mechanism that can apply an industry vertical's specific constraints
- A mechanism by which an industry vertical's terminology can be applied in the context of the horizontal (this doesn't mean that the horizontal terminology is replaced, but rather that they coexist)

To do this, what's needed is a canonical message architecture, a horizontal specification, that vertical industries can plug their information into, including additional content, constraints, and terminology. While some degree of harmonization is necessary to provide a mapping from one vertical to another, and so that common terms can be agreed on where appropriate, the complete purging of the terminology of a vertical industry is certainly neither necessary nor practical. Only by working together using a common horizontal base is it possible to achieve a canonical business language for integration (see Figure 4). A canonical business language must enable communication of information between companies as well as to the business applications inside the company that drive the business. This must be done such that the terminology that has meaning in each vertical industry is preserved; otherwise the investment in industry standards has been lost and we start over from square one.

Information must then be able to pass from vertical to vertical (V2V), business to business (B2B), application to application (A2A), and application to execution (A2E). Only by addressing all these areas is it possible to provide a canonical business language.

About Open Applications Group

The Open Applications Group is a nonprofit consortium focusing on best practices and process-based XML content for e-business and application integration. It is the largest publisher of XML-based content for business software interoperability in the world.

Technical Benefits

▶ The common model established by OAG enables the user community to adopt a single interfacing approach for all applications supported rather than separate ones for each vendor they implement.

▶ OAG can provide a forum for each of the four stakeholders, as defined by customers, application vendors, technology vendors, and systems integrators to discuss their needs and issues and be heard as a unified voice.

▶ The OAG common model enables a single mapping strategy for all multivendor implementations in the enterprise, making support and maintenance easier as technologies change.

Historical Brief

OAG was formed in 1995 by the leading enterprise software vendors. Each funded the organization and submitted volunteers to the design teams set up as the foundation organization.

The first step was to establish a common model that all of the specifications could be written to. The teams then defined the business scope that the specifications would initially include. After defining the business interface data and process requirements, and gaining agreement among the membership, the team introduced these specifications to the customer to gain user acceptance concerning the effectiveness of the specifications and their implementation by the OAG member community.

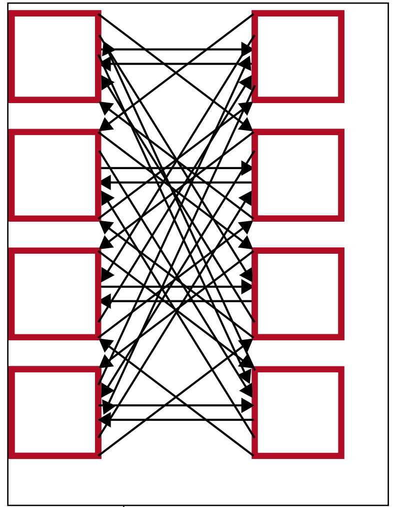


FIGURE 1 | Point-to-point integration

What's needed is a horizontal business language that is open to working with vertical industries, that doesn't require these industries to forgo the body of work they've built over the years, and that has a proven track record of delivering both B2B integration and A2A integrations.

OAGIS: A Canonical Business Language

The Open Applications Group's Integration Specification has a proven track record for B2B and A2A integrations with companies such as Agilent, Lucent, Ford, Boeing, Lockheed-Martin, and IBM, as well as support in products from vendors such as Oracle, PeopleSoft, IBM, Mercator, HK Systems, and JD Edwards.

In moving to support XML Schema, OAGI recognized through the feedback of our members that the key feature of OAGIS is its extensibility.

While OAGIS provides a good set of fields, compounds, and components needed for integration, there is always going to be something that the designers of the specification didn't consider (no matter how smart they are). Likewise, a message can often be used in business scenarios or collaborations in ways that the designers never envisioned. Additionally, through our work with industry groups such as STAR (Standards for Technology for Automotive Retail) and HR-XML, OAGI has learned that an industry vertical can often reach consensus among its members to restrict certain values, and certain collaborations can be made fixed or adhere to certain constraints within a given industry. However, when communicating across industries this doesn't hold true.

Because of these lessons learned, OAGIS 8.0 has been designed with the ability to plug in industry vertical content, constraints, and terminology through the use of an OAGIS overlay. Use of these overlays for industry verticals makes it possible for OAGIS to be a canonical business language, as shown in Figure 5.

OAGI's focus is on creating a canonical business language that enables the communication of information required to do business in order to achieve the "end-in-mind." As such, OAGI doesn't contain the in-depth knowledge of particular industries – such as automotive, human resources, pharmaceuticals, finance – that the industry groups working in these areas do. Likewise, these vertical industry groups don't have in their charters to create "canonical business language" to enable businesses to do business across vertical boundaries.

Open invitation

Because of this, OAGI maintains an open policy and invites other industry groups to partner with the OAGI to create an overlay of OAGIS that utilizes the terminology that already exists as well as the wealth of knowledge that each industry group has.

Only by working together can we create a true "canonical" business language that will allow companies to achieve the "end-in-mind."

HR-XML uses OAGIS

A real example of this is the partnership between OAGI and HR-XML. As a result of this partnership, HR-XML and OAGI have agreed to reference each other's work. For example, HR-XML has deployed their SIDES specifications in the OAGIS 8.0 architecture, making use of the overlay capability of OAGIS 8.0. This allows the HR-XML community to take advantage of the BOD constructs to identify the intent of, for example, a TimeCard, and add the additional value of the BODs to identify the transactional differences of different verbs on the TimeCard, Noun – for example, a Get transaction request information. A Show is the response to the Get that shows the information requested. The HR-XML community also can use business documents – OAGIS BODs that HR-XML doesn't have within its charter to define without worrying about different data formats and structures.

Benefits to the OAGI community are that OAGI doesn't have to define duplicate human resource BODs. And the OAGI community, like the HR-XML community, can use the HR-XML specifications without worrying about different data formats and structures.

Both organizations will reference each other's work as appropriate.

Note: Although OAGI does have a few human resources BODs, they are not to the detail that HR-XML has defined, and as such, implementations using BODs today can use the specification of HR-XML just as they would use an OAGIS BOD in the future.

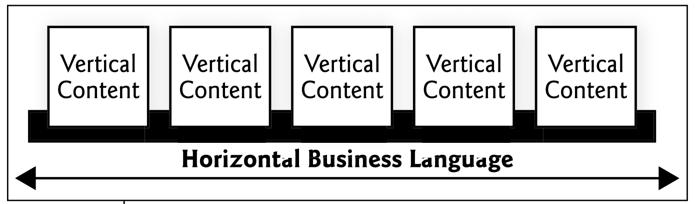


FIGURE 2 Horizontal business language that supports industry content

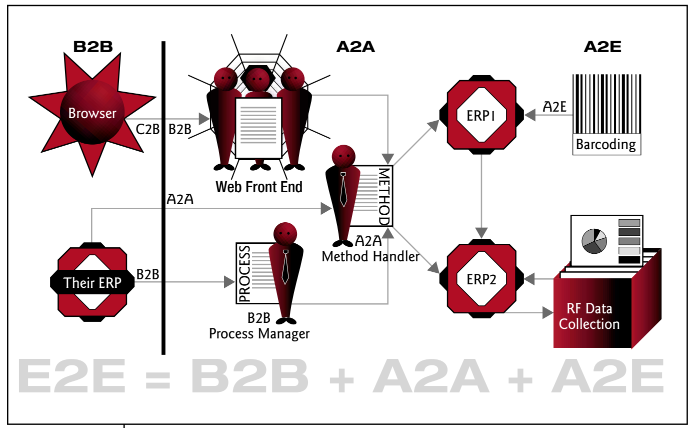


FIGURE 3 End-to-end integration

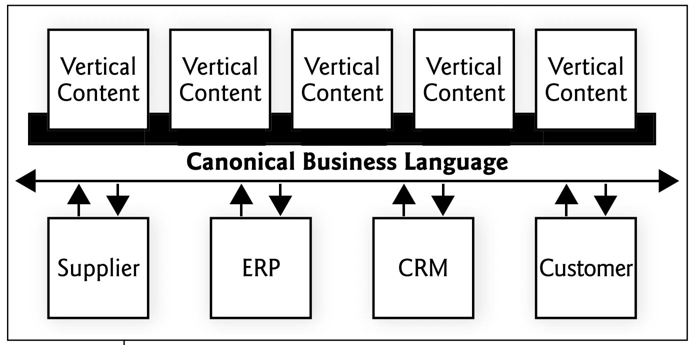


FIGURE 4 The "end-in-mind"

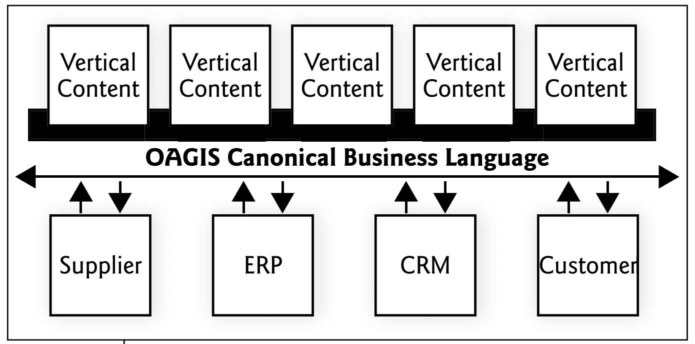


FIGURE 5 OAGIS 8 provides the end-in-mind

STAR uses OAGIS

Another real example of this is the partnership between STAR and OAGI. STAR has chosen OAGIS 8.0 as the basis for the STAR community's XML initiative. This initiative is designed to streamline the interactions between auto dealers, automotive manufacturers, and other business partners in the automotive distribution value chain.

Through this partnership with OAGI, the STAR community has built new BODs that will be made available in a future release of OAGIS. By working through this process, it has been possible to identify vertical aspects and horizontal aspects of several business messages. These have been captured at the appropriate level. OAGI has received more feedback on BODs that were being worked through to ensure that they meet everyone's needs.

STAR is also able to focus on their core competency, and that is to create standards for the automotive retail industry.

It's important to note that while OAGIS makes use of a flexible and extensible model, STAR has chosen to prohibit extension in the STAR overlay. OAGIS 8.0 enables this capability.

Summary

We have an opportunity to work together, the Open Applications Group and industry groups, to define a true canonical business model in which each industry can maintain its independence while leveraging cross-industry work to maximize interoperability while minimizing redundancy. OAGI issues this call for convergence to all industry groups to work together in a loosely coupled XML alliance for the future. For more information e-mail us at info@openapplications.org or go to our Web site at www.openapplications.org.

Acknowledgements

Thanks to the OAGIS 8 architecture team, which also includes Mark Feblowitz, Frictionless Commerce, on loan to OAGI; Andrew Warren, Canopy International, on loan to OAGI; Kurt Kanaskie, Lucent Technologies; Satish Ramanathan, MRO Software; with much help from Duane Krahn, Irista.

MROWELL@OPENAPPLICATIONS.ORG

AUTHOR BIO

Michael Rowell, chief architect for OAGI, is responsible for technical and content development for the group's specifications. He led the OAGIS 8.0 architecture team in adapting OAGIS to XML Schema, resulting in OAGIS 8.0. Michael has been involved with XML since the beginning, in his current position as well as previously with IBM, in both solution provider and tool provider roles.

CTIA WIRELESS I.T. & INTERNET 2002

www.ctiashow.com

WRITTEN BY **SURESH SELVARAJ**

Generate PDF Files with XML, XSL-FO, and FOP

A step-by-step guide for the programmer

This article will give you enough information to use the major features of XSL Formatting Objects (XSL-FO) in conjunction with Apache's FOP API for rendering documents in Adobe's Portable Document Format (PDF).

The W3C's specification for Extensible Stylesheet Language comes in two parts:
- ***XSLT:*** A language for transforming XML documents
- ***XSL-FO:*** An XML vocabulary for specifying formatting semantics

FOP (Formatting Objects Processor), which is part of Apache's XML project, is the world's first print formatter driven by XSL formatting objects. It's a Java application that reads an XSL-FO file and renders the output in PDF format. Other formats supported are XML, SVG, PS, PCL, Print, AWT, MIF, and TXT. To dig deeper, you may want to visit http://xml.apache.org/fop.

This tutorial uses Sun's JAXP API for XSLT transformation and Apache's FOP API for rendering PDF output. We'll use a Journal Subscription form that allows the user to enter details like name, payment mode, and bank details to subscribe to a journal. The form is a simple JSP page. Once the form is submitted, the request is forwarded to a servlet that captures the form details and constructs an XML string. XSL-FO stylesheet is applied to the dynamically created XML string and then transformed using JAXP API. The intermediate ".fo" file created as a result of transformation is used as input by the org.apache.fop API for rendering PDF output.

The following steps are used to create our subscription form in PDF format:
1. Create an XSL-FO stylesheet.
2. Transform the XML XSL-FO using JAXP API to produce an intermediate ".fo" file.
3. Use org.apache.fop API to convert the ".fo" file to PDF.

AUTHOR BIOS
Suresh Selvaraj works as an IT analyst for Tata Consultancy Services in India. A Sun certified Java programmer, he is involved in the development of J2EE-based projects.

Creating the XSL-FO Stylesheet

Listing 1 is the outline of a simple XSL-FO stylesheet.
- <*fo:root* is the root element.
- <*fo:layout-master-set*> encapsulates the simple-page-master.
- <*fo:simple-page-master* defines the page height, width, top and bottom margins, and left and right margins. It has a master name that is referred by the page that uses the characteristics of a page defined in the <fo:simple-page-master>.
- <*fo:page-sequence* defines the page-sequence-master and refers to the simple-page-master by using the master-reference attribute. This means that the characteristics of a page defined in the simple-page-master will be used by the page-sequence-master using the master-reference attribute. The value specified in the master-reference attribute must be the same in both the simple-page-master and the page-sequence-master. You may also specify the order in which a given simple-page-master will be used by the page-sequence-master.
- <*fo:flow* defines the contents of a page that flows into the xsl-region-body.
- <*fo:block* defines the block where the actual contents that appear in a PDF document are placed. Each fo:block prints the contents on a new line. The fo:block can contain <fo:inline elements. You may also place tables within an fo:block.

To place the contents in each column of a table, insert the fo:block element in the fo:table-cell element as shown in Listing 2.
- ***<fo:table:*** Equivalent to an HTML <TABLE> tag
- ***<fo:table-column:*** Defines the number of columns used in a table – here, one
- ***<fo:table-row:*** Equivalent to an HTML <TR> tag
- ***<fo:table-cell:*** Equivalent to an HTML <TD> tag

You may nest tables within tables similar to what we do in HTML. To place another table in the preceding example, insert a complete set of <fo:table></fo:table> inside the <fo:block></fo:block>. The Subscription.xsl in our case study uses nested tables, as seen in Listing 3.

A stylesheet now needs to be applied to each element in the XML String document that is constructed dynamically in the servlet. For the full source code of Subscription.xsl see the note at the end of this tutorial.

Using the org.apache.fop API

As mentioned earlier, the XML file will be generated dynamically using the values entered in the Subscription.jsp form. Subscription.xsl is applied to the XML document that's created dynamically and transformed using JAXP API.

Import the following org.apache.fop classes in XSLTOPDFServlet:

```
import org.apache.fop.messaging.
 MessageHandler;
import org.apache.fop.apps.Driver;
import org.apache.fop.apps.*;
import org.apache.log.*;
```

Using the JAXP API

The "Subscription.fo" file in Listing 4 created in the previous step (foFile) is used to create the PDF document using the fop API in Listing 5. The complete source code and Subscription.pdf can be downloaded from www.sys-con.com/xml/source.cfm. You can download org.apache.fop API from http://xml.apache.org/fop.

I tested this application using BEA WebLogic Server 6.1. For instructions on how to set up and run the example, please refer to the README file included in the zip.

If you don't have access to a Web server, download the FOP API and run the standalone Java program that comes as part of the fop API download. For instructions, please refer to the README file included in the zip.

SSELVAR1@YAHOO.COM

LISTING 1

```
<xsl:stylesheet version="1.0"
 xlmns:xsl="http://www.w3.org/1999/XSL/Transform"
xlmns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:template match="page">
<xsl:processing-instruction   name="cocoon-format">type=
 "text/xslfo"</xsl:processing-instruction>
<fo:root xlmns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
<fo:simple-page-master master-name="tss_Page1"
  page-width="297mm"  page-height="240mm"
  margin-top="0.5in"  margin-bottom="0.5in"
  margin-left="0.5in" margin-right="0.5in" >
<fo:region-before extent="5.0in"/>
<fo:region-body margin="50mm 00mm 50mm
  0mm"/>
<fo:region-after extent="1.0in"/>
</fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference ="tss_Page1">
<fo:flow flow-name="xsl-region-body">
<fo:block text-align="center"> Hi There!
</fo:block>
<fo:block text-align="center"> Hello World!
</fo:block>
</fo:flow>
</fo:page-sequence>
</fo:root>
</xsl:template>
```

LISTING 2

```
<fo:table  border-spacing="3in" text- align="center">
<fo:table-column column-number="1" column-
width="200mm" />
<fo:table-body>
<fo:table-row text-align="left">
  <fo:table-cell border-style="solid" border-
    width="0mm" background-color="#FFFFFF">
    <fo:block text-align="center">
        Hello World!
   </fo:block>
 </fo:table-cell>
</fo:table-row>
      </fo:table-body>
</fo:table
```

LISTING 3

```
<xsl:stylesheet version="1.0"
 xlmns:xsl="http://www.w3.org/1999/XSL/Transform"
 xlmns:fo="http://www.w3.org/1999/XSL/Format">
<xsl:template match="page">
 <xsl:processing-instruction  name="cocoon-format">type=
  "text/xslfo"</xsl:processing-instruction>
<fo:root xlmns:fo="http://www.w3.org/1999/XSL/Format">
<fo:layout-master-set>
       <fo:simple-page-master
          master-name="tss" page-width="297mm"  page-
          height="240mm" margin-top="0.5in"  margin-
          bottom="0.5in" margin-left="0.5in" margin-
          right="0.5in" >
          <fo:region-before extent="5.0in"/>
          <fo:region-body margin="50mm 00mm 50mm
           0mm"/>
           <fo:region-after extent="1.0in"/>
     </fo:simple-page-master>
</fo:layout-master-set>
<fo:page-sequence master-reference ="tss">
     <fo:flow flow-name="xsl-region-body">
     <fo:table  border-spacing="3in" text-
        align="center">
        <fo:table-column column-number="1" column-
          width="200mm" />
        <fo:table-body>
        <fo:table-row text-align="left">
          <fo:table-cell  border-style="solid" border-
             width="0mm" background-color="#FFFFFF">
             <fo:block text-align="center">
   <fo:table border="1pt solid black" >
       <fo:table-column column-number="1"
                             column-width="72mm" />
      <fo:table-column column-number="2"
                             column-width="120mm" />
                           <fo:table-body>
                    <xsl:apply-templates></xsl:apply-templates>
                           </fo:table-body>
                      </fo:table>
                 </fo:block>
                </fo:table-cell>
            </fo:table-row>
          </fo:table-body>
      </fo:table>
    </fo:flow>
</fo:page-sequence>
</fo:root>
</xsl:template>
```

LISTING 4

```
// Create an instance of TransformerFactory class.
TransformerFactory  tfactory=
 TransformerFactory.newInstance();

// Create an instance of Transformer class.
// The input to Transformer instance is the xsl file.
// Here xslFilePath refers to the absolute file path of
// Subscription.xsl file.

Transformer transformer=
 tfactory.newTransformer (
 new StreamSource(xslFilePath));

// Transform the dynamically created xml file
// (xmlString) which is of type java.lang.String.
// Output is an intermediate ".fo" file, Subscription.fo.
// [ "foFile" should refer to the absolute path of
// Subscription.fo file ]

transformer.transform (new StreamSource(xmlString),new
StreamResult(foFile));
```

LISTING 5

```
// Create an InputSource that takes the
// "Subscription.fo" file as input.

InputSource fopfile = new InputSource(new
FileInputStream(foFile));

Invoke the method
renderPDF(InputSource in, OutputStream out);

public static void renderPDF (InputSource in, OutputStream
out) throws Exception
{
//Set the parser type.
  System.setProperty("org.xml.sax.parser",
  "org.apache.xerces.parsers.SAXParser");

// Instantiate the org.apache.fop.apps.Driver class
// and pass the InputSource and Servlet
// OutputStream objects.
 Driver driver = new Driver(in, out);
driver.setLogger(log);

// Set the renderer type to PDF.
driver.setRenderer(Driver.RENDER_PDF);

driver.addElementMapping("org.apache.fop.fo.StandardElement
Mapping");
driver.addElementMapping("org.apache.fop.svg.SVGElementMapping");

// Invoke the run() method.
driver.run();

// Flush the OutputStream.
out.flush();
}
```

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

XML NEWS

Contracts Release 3.0 New from diCarta

(Long Beach, CA) – diCarta has unveiled the newest version of its Enterprise Contract Management Solution. The release introduces advanced contract assembly and template support, an enhanced workbench for contract authoring and collaboration, and expanded XML integration.
www.dicarta.com

Skywire Software Upgrades Object Manager

(Frisco, TX) – Skywire Software has announced the availability of Object Manager version 3.4. Designed to automate the transport management process, Object Manager provides functionality for monitoring, tracking, and improving change management within SAP. It also provides the capability to "undo" a transport.

New features include added uploading and downloading of transport request files; added functionality to consider transport requests with the name pattern "+++K9*" for automatic external attribute assignment and import approval; and modified configuration dialog to allow a return to previous settings if the new one cannot be saved due to a remote system being unavailable.
www.skywiresoftware.com

OASIS Forms New XML Security Group

(Billerica, MA) – The OASIS standards group has created a new technical committee to advance XML security standards, including WS-Security, which recently moved into OASIS. The committee will have its first meeting early this month, hosted by Sun Microsystems.

OASIS is working on several key security areas, including SAML for authentication and authorization, XACML for access control, XrML for rights management, SPML for exchanging provisioning information, and XCBF for describing biometrics data.
www.oasis-open.org

Software AG Appoints Industry Veterans

(Reston, VA) – Three industry veterans have joined Software AG, Inc., to lead key sales and marketing initiatives. Haskell Mayo has been named senior vice president of sales and marketing; Mike McDowell, vice president, business development and channels; and Michael Gadow, director of content management product sales.

In addition, JP Morgenthal, coeditor-in-chief of *XML-Journal,* has joined the company's Professional Services group as chief services architect. Morgenthal is an internationally prominent authority on XML with more than 15 years of experience designing, developing, and analyzing software and technology. In his new role Morgenthal will explore and manage the design of complete professional services solutions based on Software AG technology and partner products in existing and emerging industries.
www.softwareagusa.com

Vultus Launches Browser Application Platform

(Orem, UT) – WebFace Solution Suite, a new product designed to accelerate the delivery of enterprise applications over the Web, is now available from Vultus, Inc.

Consisting of the WebFace Browser Application Platform and WebFace Studio, the suite delivers an enhanced end-user experience, providing a Web-based presentation layer that mirrors the traditional desktop computing environment and removes many of the obstacles associated with static HTML Web pages. Utilizing widely accepted industry standards (XML, JavaScript, and HTML), the suite renders full-featured application windows inside a Web browser, complete with grids, tab folders, directory tree views, drag-and-drop capabilities, and other familiar components.

WebFace doesn't rely on third-party plug-ins, and doesn't reside locally on the end user's machine.
www.vultus.com

New Leaders at DataPower

(Cambridge, MA) – DataPower Technology, Inc., has appointed Steve Kelly as chairman and interim CEO, and Bill Tao as vice president of engineering.

Kelly's responsibilities will include executing the company's business strategy, promoting strategic alliances with vendors and partners, and managing internal operations. Tao is responsible for general engineering management, quality assurance, customer technical support, release engineering, and documentation.
www.datapower.com

Xbridge Host Data Connect Now Supports XML

(Los Gatos, CA) – With new support for XML and enhanced data-access and reporting capabilities, Xbridge Host Data Connect now makes it easier to use mainframe data in virtually all key desktop applications. Host Data Connect enables Windows and Web-enabled applications to directly access mainframe data and to see that data in the format of the requesting application, all in real time.

In addition, the Host Data Access Utility supports the creation of text files (for use with virtually any text editor), delimited files (for spreadsheets), and Microsoft Access databases. Host Data Connect continues to support Microsoft Word, Excel,

XMLEdge 2002 West Conference & Expo Coming October 1-3

Educational Value of the Year

(Montvale, NJ) – **XMLEdge 2002 West**, colocated with **Web Services Edge 2002 West**, will take place October 1–3 at the McEnery Convention Center in San Jose, California. **SYS-CON Events**, in an unprecedented move, has lowered registration prices to help developers, architects, programmers, and *i*-technology professionals make the critical exchange of information that will bring their careers to the next level.

Over 60 comprehensive conference sessions organized into five tracks – XML, Web Services, Java, IT Strategy, and Wireless – represent the best opportunity to hear from industry leaders and visionaries the latest on tools and strategies to implement Web services and integrate applications. Leading industry keynotes include **Jonathan Schwartz**, *Sun Microsystems*; **Kaj van de Loo**, *SAP*, and **Barry Morris**, *IONA*. Register online at www.sys-con.com.

To make the most of the conference, and to get up to speed on Web services before the fact, attend the **Web Services Edge World Tour** at the Westin Bonaventure in Los Angeles on September 19. Register at www.sys-con.com/education/.

OWNED BY SYS-CON MEDIA
PRODUCED BY SYS-CON EVENTS

XML NEWS

and Web formats. An improved results template makes searches across all data types faster, easier, and more targeted.
www.xbridgesystems.com

NetReflector Upgrades InstantSurvey

(Seattle) – InstantSurvey 4.0 by NetReflector, Inc., has been designed to meet the complex online survey needs of enterprise users.

Based on open XML standards, InstantSurvey 4.0 enables distributed workgroups to collect massive amounts of information in a short period of time and instantly gain intelligence on feedback results.

In addition to a comprehensive list of standard graphical reports, InstantSurvey 4.0 introduces the flexibility to easily build custom reports, create executive-level presentations, or perform interim data downloads to other applications, such as spreadsheets and databases, for further analysis.
www.netreflector.com

HR-XML Expands Benefits Enrollment Standard

(Raleigh, NC) – The HR-XML Consortium has approved an updated version of its XML specification for employee benefits plan enrollments. Version 2.0 of the specification supports enrollment and maintenance of human resources in (1) tier-based coverages such as medical, dental, and vision; (2) spending accounts, commonly known as flexible spending accounts (FSA); and (3) rate-based coverage such as life and short- and long-term disability. Version 2.0 also includes built-in "user areas" to accommodate trading-partner customizations.

Register to download the freely available enrollment specification at www.hr-xml.org/forms/schema_register.cfm.

RosettaNet Merges with Uniform Code Council

(Lawrenceville, NJ) – Under an agreement reached by the Uniform Code Council, Inc. (UCC), and RosettaNet, the latter will become a subsidiary of the UCC while continuing to operate as an entity directly with its members.

A primary goal of the relationship is to accelerate B2B integration via industry implementation efforts for XML standards and emerging services. The UCC and RosettaNet have worked together since 1998, collaborating on existing projects to promote a single, unified approach to the standards adoption process.
www.uc-council.org/
www.rosettanet.org/

AdventNet Releases Middleware Manager 4.0 WebLogic Edition

(Pleasanton, CA) – AdventNet, Inc., a leading provider of standards-based J2EE management solutions, has announced the immediate availability of AdventNet Middleware Manager 4.0 WebLogic Edition, optimized for advanced JMX (Java Management Extensions) based management of entire Web application ecosystems.

The product has a number of new features such as end-to-end Web transactional views, a root cause analyzer, and an extensible GUI.
www.adventnet.com

Software AG, Altova Offer Integrated Project Bundle

(Reston, VA / Beverly, MA) – Software AG and Altova, Inc., have jointly announced the availability of the XML Spy 4 Suite and Tamino integrated product bundle, a comprehensive solution for the development of high-performance XML-based applications. The offering includes Altova's XML Spy 4 Suite and Software AG's Tamino XML Server (limited version).

The product bundle runs on all standard Windows platforms.
www.altova.com
www.softwareagusa.com

Sybase PowerBuilder 9.0 Now in Beta

(Dublin, CA) – The beta version of PowerBuilder 9.0 is now available from Sybase. This new version of Sybase's rapid application development (RAD) tool supports a number of key capabilities, including XML DataWindow, Java Server Pages (JSP), and third-party application servers. Key capabilities to be included in future PowerBuilder 9.0 beta products include Web services, XML services, and support for Microsoft .NET.
www.sybase.com

Simpler Way to Analyze, Deliver Financial Documents?

(New York / Redmond, WA) – The Nasdaq Stock Market, Inc., Microsoft Corp., and PricewaterhouseCoopers (PwC) have teamed up to demonstrate the power of XBRL (Extensible Business Reporting Language), an XML-based platform developed for corporate reporting over the Internet.

The three organizations have joined forces in a pilot program to showcase XBRL's ability to provide companies with an easier means to communicate financial information and deliver it to investors with enhanced capability to analyze data. The pilot program, designed by PwC and stored on Nasdaq hardware, provides access to XBRL data through Microsoft Office. It is available to the public at www.nasdaq.com/xbrl.

Integration Platform for Barcelona Port Logistics

(Wilton, CT) – Mercator Software, Inc.'s Data Exchange Solution has been selected by the Port of Barcelona as the integration platform for its port logistics system. During 2001 the Port of Barcelona, the principal transportation and services hub in Catalonia, handled close to 32 million tons of traffic, more than 1.4 million containers, and 1.44 million passengers. Mercator's EDI and XML integration software will enable the Barcelona Port Authority, which manages all of the port's activities, to automate the flow of information relating to harbor traffic.
www.mercator.com

Nimble Integration Suite 2.0 Available

(Seattle) – Nimble Technology, Inc., has released Nimble Integration Suite 2.0, the first data and information integration platform to provide both Open Database Connectivity (ODBC) and XQuery access.

Nimble's software gives enterprises a tool to integrate and access multiple data sources, inside and outside the firewall, for real-time data integration and information sharing capability.

The Nimble Integration Suite 2.0's new features also include enhanced concordance support, access to LDAP data, and front-end reporting.
www.nimble.com

XML-J ADVERTISER INDEX

ADVERTISER	URL	PHONE	PAGE
ADOS Co., Ltd.	http://www.a-dos.com	81-3-5475-1551	4
Altova	www.xmlspy.com		52
BEA	dev2dev.bea.com/useworkshop		2
CTIA Wireless I.T. & Internet 2002	www.ctiashow.com		43
engindata Research	www.engindata.com	201-802-3082	29
IBM	ibm.com/websphere/ebaydev		51
JDJ Store	www.jdjstore.com	888-303-JAVA	25
Macromedia	www.macromedia.com/goCFMXlight	888-939-2545	9
Mongoose	www.portalstudio.com		15
NET Developer's Journal	www.sys-con.com/dotnet	888-303-5282	19
PolarLake	www.polarlake.com		13
Sonic Software	www.sonicsoftware.com/websj		6
SYS-CON Events	www.sys-con.com	201-802-3069	33, 34
SYS-CON Media	www.sys-con.com/suboffer.cfm	888-303-5282	47
WebLogic Developer's Journal	www.weblogicdevelopersjournal.com	888-303-5282	35
WebServices Edge World Tour	www.sys-con.com	201-802-3069	21
WebSphere Developer's Journal	www.sys-con.com/websphere	888-303-5282	47

General Conditions: The Publisher reserves the right to refuse any advertising not meeting the standards that are set to protect the high editorial quality of ***XML-Journal***. All advertising is subject to approval by the Publisher. The Publisher assumes no liability for any costs or damages incurred if for any reason the Publisher fails to publish an advertisement. In no event shall the Publisher be liable for any costs or damages in excess of the cost of the advertisement as a result of a mistake in the advertisement or for any other reason. The Advertiser is fully responsible for all financial liability and terms of the contract executed by the agents or agencies who are acting on behalf of the Advertiser. Conditions set in this document (except the rates) are subject to change by the Publisher without notice. No conditions other than those set forth in this "General Conditions Document" shall be binding upon the Publisher. Advertisers (and their agencies) are fully responsible for the content of their advertisements printed in ***XML-Journal***. Advertisements are to be printed at the discretion of the Publisher. This discretion includes the positioning of the advertisement, except for "preferred positions" described in the rate table. Cancellations and changes to advertisements must be made in writing before the closing date. "Publisher" in this "General Conditions Document" refers to SYS-CON Publications, Inc.

XML Schema Best Practices

Better practices, perhaps?

BY TOM GAVEN

Tom Gaven has authored over 30 courses in many different technologies, as well as MindQ's Developer Training for Java program. In the last two years he has architected and developed products with XML, XSLT, XML Schema, RELAX NG, Java, and Schematron.

In the June issue of ***XML-Journal*** I mentioned that we need a set of best practices that rein in the complexities of XML Schema. The set offered at www.xfront.com is a great start, but they cater to the XML Schema extremists, and I'd like to modify them, offering some alternative best practices for "the rest of us."

You'll have to refer to www.xfront.org/BestPracticesHomepage.html to get a full description of the issues discussed below. Following are some ground rules I used to build my "modified" best practices list:

- **The Over 10 Page Rule (acronym: O10P Rule):** *Any "Best Practice" that takes more than 10 pages to describe shouldn't be a best practice.*
- **The Safe and Sane Use of Namespaces Rule (acronym: SASUONS Rule):** *This rule is applied as needed to maintain the sanity of the schema developer with respect to the use of namespaces.*

Best Practice #1
***Issue**: When should a schema be designed to hide (localize) within the schema the namespaces of the elements and attributes it is using, versus when should it be designed to expose the namespaces in instance documents?*

Well, this best practice wins a prize, in that it triggers BOTH the O10P rule AND the SASUONS Rule!

Conclusion: *Always* use elementFormDefault="qualified" (and attributeFormDefault="unqualified") in your schemas. It's the *only* sane way to go.

Best Practice #2
***Issue**: When should an element or type be declared global, versus when should it be declared local?*

Here's a case where XML Schema gives us too many choices, making it too confusing, without really giving any bang for the buck. My recommendation is to *always* declare elements (and attributes) locally, and *always* declare types globally. The exception is that root element *must* be declared globally.

Best Practice #3
***Issue**: When should an item be declared as an element versus when should it be defined as a type?*

This best practice needs to be removed from the list. Elements and types are disjoint schema components. You need to declare an element when you need to declare an element! That is, an element declaration is needed for every element found in the instance document. You need to declare a type when you need to declare a type. You need to declare a type when you are constructing content models.

Best Practice #4
***Issue**: In a project where multiple schemas are created, should we give each one a different targetNamespace, or should we give all the schemas the same targetNamespace, or should some of them have no targetNamespace?*

This definitely triggers the SASUONS Rule, just by the description above.

Conclusion: Give each vocabulary a separate targetNamespace, except when you benefit by breaking down a very large vocabulary into multiple physical schema documents.

Best Practice #5
***Issue**: What's the best practice for implementing a container element that's to be composed of variable content?*

This best practice triggers the O10P Rule. Again, XML Schema is just too complex. I'm against using substitution groups, abstract elements, xsi:type, and complexType inheritance. I just think they add too much confusion to schema development, and aren't worth the pain.

Conclusion: Go with the proposed method 2, the <choice> element – and I'd throw in the liberal use of model groups. Container elements can be created that reference (reuse) model groups, combined together via the <choice> element.

Best Practice #6
***Issue**: Should you design your schemas to build type hierarchies (design by subclassing), or should you design them to aggregate components (design by composition)?*

I agree with the conclusion at xfront.org here: design by composition is the preferred approach. However, I'd add a recommendation to use XML Schema model groups as the preferred way to design a content model via composition. ComplexType inheritance is overused and broken. Use simpleType as needed.

Best Practice #7
***Issue**: What's the best practice for creating extensible content models?*

I like the question, but don't like the answers proposed at xfront.com for this one. There are two proposals: (1) use complexType inheritance and xsi:type, and (2) use the <any> element. I don't like either complexType inheritance or xsi:type. Use of the <any> element, as discussed at xfront.com, has nondeterministic problems.

Conclusion: Use XML Schema Model (and Attribute) groups for creating extensible content models. These offer extensible content models for elements and attributes without any of the problems associated with complexType inheritance or xsi:type.

• • •

These modified best practices should enable you to design more understandable XML Schemas.

TGAVEN@XMLDISTILLED.COM

IBM
ibm.com/websphere/ebaydev

ALTOVA

www.xmlspy.com