

XML Collaborator

XML Design Collaboration
and Registry Software

CONTENTS

Background.....	3
Working together: Collaboration and the design process	3
Making collaboration work	4
Distributing the products of your work: Registries	5
What registry standards already exist?	6
A comprehensive approach to an XML registry	8
Combining collaboration and registry: The XML Collaborator solution.....	10
XML Collaborator architecture	11
XML Collaborator features	12
Conclusion	15

BACKGROUND

One of the most challenging problems faced by enterprises and trading networks today is the reliable exchange of information. Industry consortia in vertical markets such as mortgages, credit reporting, pharmaceuticals, and automotive have convened with a single goal in mind: to define how they can share information with one another in a reliable, reproducible fashion.

Over the years, many different solutions to the information reuse problem have been attempted—from the exchange of flat-format files on tapes to more sophisticated EDI interchange of information. Recently, however, two technologies have emerged that may provide one of the best solutions to this problem: XML and Web services.

The eXtensible Markup Language, or XML, allows structured information to be represented in a standardized, programmatically-accessible fashion. The XML solution for the representation of information is superior to flat files because XML allows the internal structure of information to be easily represented, avoiding traditional flat file problems such as repeating blocks. Moreover, XML is an industry standard developed by leading computing technology companies such as IBM and Microsoft, and is readily and freely available to any company that wants to adopt XML as a data representation platform.

Web services, or XML Web services as they are sometimes known, provide a way for systems to exchange XML information over the Internet. XML documents are associated with operation information, creating a message: “Here’s some information, and here’s what to do with it.” Using Web services, companies can set up sophisticated request-response mechanisms, as well as other types of messaging such as simple one-way messages. The combination of XML and Web services allows systems to easily interact with each other, working in concert to perform enterprise-class functions.

As XML and Web services become more pervasive, however, companies and consortia that want to design these enterprise-class behaviors face two hurdles. The first is the collaboration hurdle: how are the XML documents and Web service interfaces for a particular set of inter-process or inter-company requirements structured, and how can companies work together to create these documents and interfaces? The second is the registry hurdle: once the documents and interfaces are designed, how can they be exposed to participants that want to take advantage of those interfaces?

WORKING TOGETHER: COLLABORATION AND THE DESIGN PROCESS

There are typically many different stakeholders in the design process for standards that are created for the enterprise-level sharing of information between systems or companies. Each system may have slightly different requirements or intentions for the standard, and each stakeholder will want to ensure that their needs are met by the resulting structures. To produce the ideal result, then, there needs to be some way that the stakeholders can **collaborate** on the design process.

Collaboration is not an insignificant task. For many development efforts, the stakeholders will not necessarily be based out of the same facility.

Historically, collaboration efforts have involved the stakeholders (or their representatives) boarding planes and traveling to some common meeting location to hash out the specifics of a particular design. This is an incredibly costly way to build these structures, both in terms of the disruption in the stakeholders' schedules and the travel costs incurred. Moreover, there will often be issues that cannot be easily resolved in a day or two—the appropriate subject matter expert may not be at the meeting, or the stakeholder may need to review a particular issue with his or her team.

Another concern when collaborating on document design is that the end result—the XML Schemas, the Web service WSDL specifications, or whatever other outcome is desired—accurately reflects the input of all the participants in the design process. Because these structures are simply flat files, a danger exists that changes may be lost—if two different groups make a change to a particular schema, for example, those changes will need to be integrated together in the final result. Tracking these changes and ensuring that a good paper trail exists to resolve who made a change and when is a crucial part of a strong collaboration process.

Currently available XML design tools provide little or no true collaboration functionality; most groups that are attempting to design XML structures for information sharing resort to the exchange of spreadsheets of data points with no version control and no ability to consistently indicate what changes have been made and by whom.

Making collaboration work

Collaboration can be facilitated by the use of a collaboration tool—a piece of software that allows participants in the design process to work together in a virtual environment, reducing the need for travel and face-to-face meetings. The collaboration tool should allow users to track how the various components in a particular structure are being changed and by whom, as well as the various versions of components (to allow the designed structures to be versioned over time). The collaboration tool should also allow users to focus on the reuse of individual data structures, rather than the specific design of XML Schema structures or Web service interfaces, as seen in Figure 1.

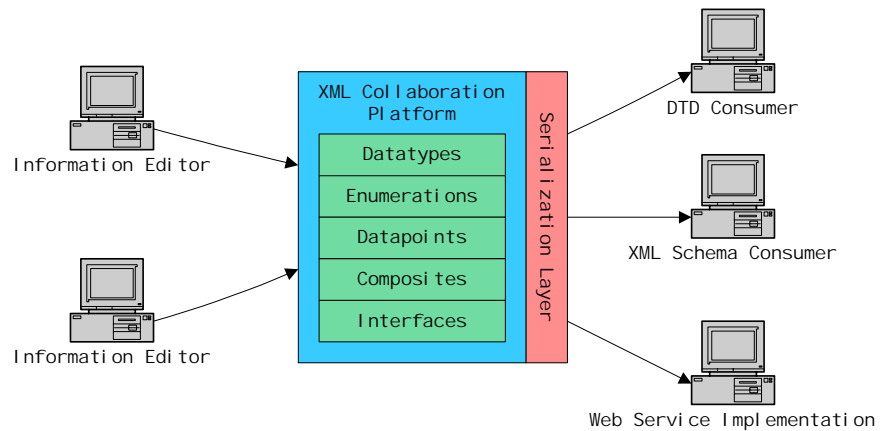


Figure 1: Collaboration platform

In this example, information editors (structure designers) work together to create the building blocks of the structures being created—datatypes, enumerations, datapoints, and so on. These building blocks are then associated together into higher-level structures such as composites and interfaces. As users want to adopt these structures in their own development efforts, they may then build the structures into the appropriate form—Web service specifications (WSDL), XML document structures (XML Schema or DTDs), or any other form the platform supports. By abstracting the design of the elements and structures away from any particular view of those structures, the design can be easily reused.

DISTRIBUTING THE PRODUCTS OF YOUR WORK: REGISTRIES

The results of a successful information design collaboration process are typically stored in some sort of registry for distribution. A **registry** is a common location where metadata about specific data elements, structures, and/or service interfaces can be registered by companies that want to share those structures with others. Interested parties can examine the metadata in the registry to determine if a particular information item exists that meets their data or communication needs. Figure 2 provides an example of an XML registry. XML document creators design XML Schemas—descriptions of allowable content in a particular XML document—and register them in a centralized location; companies or organizations that then want to create documents based on those schema can search the registry for the appropriate schema, extract them, and build documents from them.

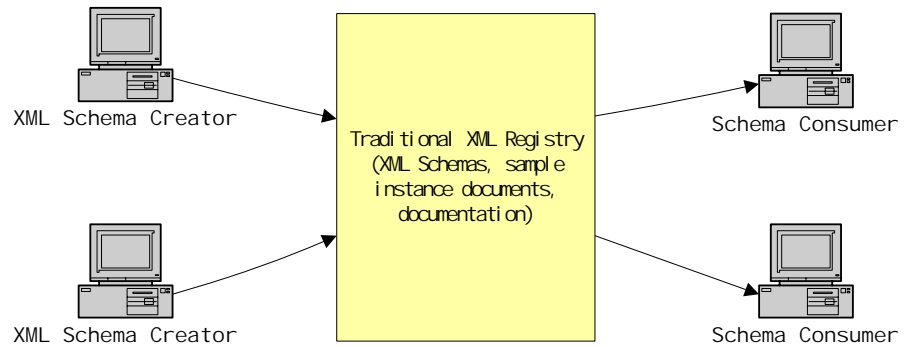


Figure 2: A traditional XML registry

A registry should track the company or companies responsible for the development of a particular schema, as well as descriptive information about that schema and the structures that comprise it. It should also be programmatically accessible, so that schemas can be extracted from it as necessary by companies that want to leverage those schemas in their software architecture.

What registry standards already exist?

There are already several registry standards published with standards bodies. These standards provide a set of approaches that may be used to help standardize individual data points, structures, and services.

ISO 11179

The ISO 11179 standard (www.iso.ch), finalized in June of 1998, provides some guidelines for the description of individual data elements. The ISO 11179 standard is broken into six parts:

- *ISO 11179-1 Framework for the specification and standardization of data elements*: Provides an overall view of how data elements should be annotated and classified. It basically serves as a roadmap to how the other parts of the ISO specification should be used.
- *ISO 11179-2 Classification for data elements*: Defines an ontological classification mechanism for the clear and unambiguous classification of individual data elements (values).
- *ISO 11179-3 Basic attributes of data elements*: Provides a brief list of metadata that should be collected for data elements. These include information about the value itself (such as whether it is required or optional, how many times it may appear, and what the allowable values for the element are) as well as information about the meaning of the element (representational information from ISO 11179-2, keywords, and so on).

- *ISO 11179-4 Rules and guidelines for the formulation of data definitions:* Provides some best practice naming and definition approaches for individual data elements. This includes simple rules such as “element names should be singular”, as well as more sophisticated guidelines such as “element definitions should not contain circular references.”
- *ISO 11179-5 Naming and identification principles for data elements:* Provides some standardized mechanisms for the unique identification of data elements. It includes some guidelines for creating element names from the classification of those elements, as well as a mechanism for associating a universally unique identifier with each element.
- *ISO 11179-6 Registration of data elements:* Describes a methodology for the registration of individual data elements. It describes how data elements should be versioned over time, and provides for a way to uniquely identify a data element through a combination of its registry authority identifier, its own unique identifier within that registry, and its version number.

While the ISO 11179 standard provides an approach to the ontological classification of data elements, as well as a mechanism to define the allowable value domain for each element, by itself it does not provide enough information to create a robust XML registry. No formal mechanism is included for the association of data elements into structures or interfaces, and the informal mechanism does not contain a rigorous way to retrieve associated elements programmatically.

ebXML

The OASIS standards body is responsible for the Electronic Business using eXtensible Markup Language, or ebXML, standard (www.ebxml.org). This standard actually consists of several more granular standards, including a standard for a collaboration profile describing an agreement between trading partners, a messaging service similar to the Simple Object Access Protocol (SOAP), a business process specification schema, and a registry information model. This registry model was promoted to version 2.0 in December of 2001, and provides a common mechanism for the sharing of technical information (XML Schemas), business process descriptors (BPSS documents), and collaboration partner profiles and agreements (CPP/A documents) to fully describe the business relationships between two or more participants in an information sharing effort. Because the ebXML registry effort is a natural outgrowth of ebXML's other goals (including the BPSS documents and CPP/A documents), an ebXML registry is both a technical registry (so-called “green pages”) as well as a business registry (so-called “white” and “yellow” pages).

The ebXML registry standard is focused on the sharing and reuse of XML Schema documents. While it provides extremely robust business registry functions, the technical registration of ebXML does not have a granular enough scope to promote the true reuse of data items and structures.

UDDI

The Universal Description, Discovery, and Integration specification, or UDDI, (www.uddi.org) provides a dynamic registry for Web services. Businesses that wish to register their service offerings with a UDDI registry may specify metadata about their company, as well as business (descriptive) and technical (binding and definition) metadata about their service offerings.

UDDI's registry model is specifically geared to the description of Web services. More granular information, such as the metadata for individual structures within the Web service messages or the metadata for data points within those structures, cannot be provided through the UDDI registry.

A comprehensive approach to an XML registry

As we've seen, ISO 11179 provides a mechanism for the registration of individual data elements; ebXML's registry model describes a registration strategy for XML Schema; and the UDDI specification details the registration of Web services. However, these functions do not stand alone—they are actually interrelated. XML Schema are built up out of individual data points, and Web services build upon those schema.

In an ideal registry implementation, all of the metadata for data elements, XML Schema, and Web services should draw from the same registry. Otherwise, data elements and XML Schemas will need to be registered multiple times (either on their own or as part of other structures), leading to potential synchronization errors as a particular enterprise metadata collection effort grows and changes over time.

Granular reuse of information design elements

A robust registry will allow specific data elements, structures, datatypes, and enumerations to easily be reused across the enterprise. The registry specifications that exist to date do not directly provide support for the reuse of individual data structures (XML "elements") and data points (XML "text elements" or "attributes") outside of the context of an XML Schema. The registry should allow these information components to be individually tracked and versioned, and any other part of the registry should be able to leverage these components in other structures.

Peer-to-peer distributed server environments

In a true enterprise-class solution, there may be thousands of different participants in the information design and registry process. In many cases, there will be a certain level of overlap in the information needs of the various participating entities; on the other hand, many information items will be proprietary to a particular stakeholder. There are three approaches to this problem. In the first approach, a single system is used to manage all of the various information items across the enterprise.

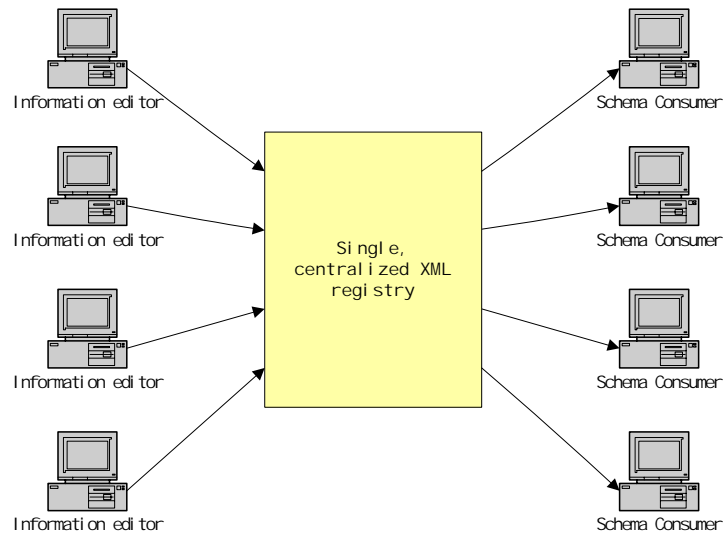


Figure 3: Centralized XML registry

The danger with this approach is obvious. In this topology, a single system must handle the information description needs of every participant in the design and registration process. This system now becomes a single point of failure for the entire data standardization effort. In addition, because this single system must manage all of the metadata for the standardization, the load on this system can easily become too great for it to bear as the enterprise information modeling effort continues to grow in scope.

Another approach is to keep the various information design efforts separate and distinct, with each group of stakeholders maintaining its own system.

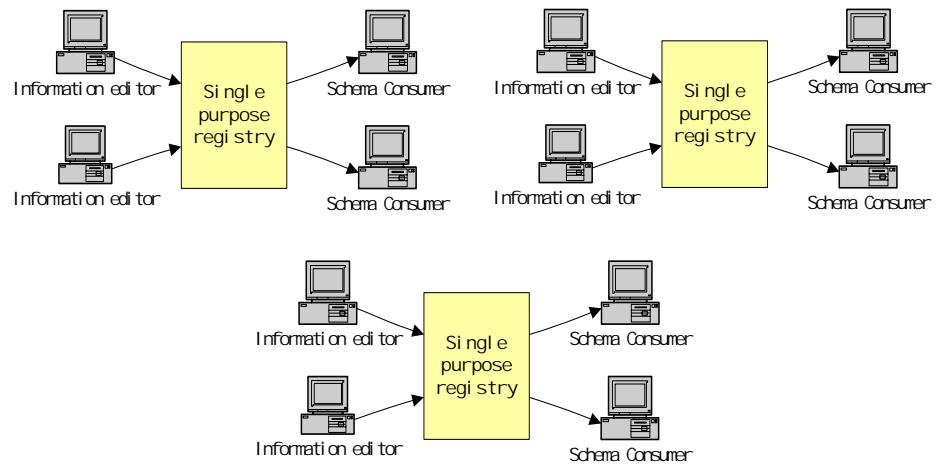


Figure 4: Separate XML registries

In this case, the single point of failure problem is solved, but now sharing information across the systems becomes much more difficult. As an information model changes in one system, that change needs to somehow

be propagated to every other system that wants to share that particular information model. This exposes the metadata to the risk of propagation errors (due to improperly propagated changes to the information model), as well as reducing the likelihood that particular components of the information model will be identified as similar or synonymous thus decreasing reuse.

The third approach is to allow multiple registries to operate in a peer-to-peer fashion.

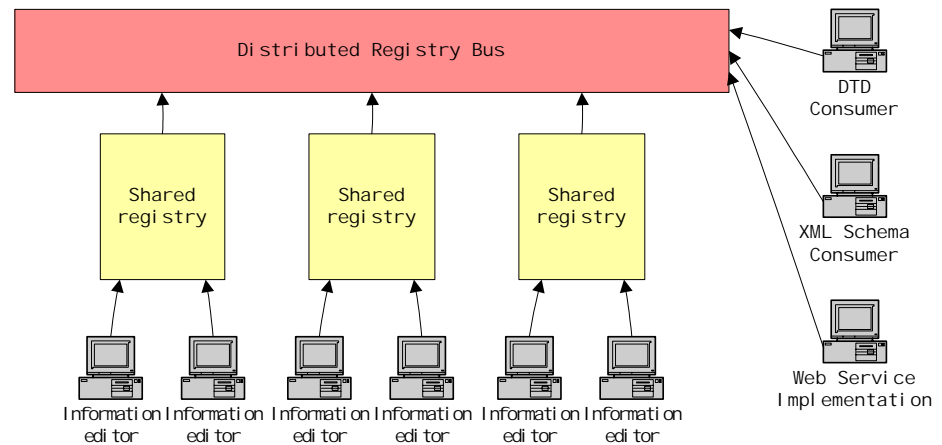


Figure 5: Peer-to-peer XML registries

In this model, the single point of failure problem is eliminated, and the information is shared freely between the systems. Information composites (built up from other information composites or datapoints) can pull their children from any server that participates in a particular networked effort, making it easy to create documents that leverage existing work from many different sources. In addition, each group of stakeholders will be responsible for its own system, distributing the cost of the unified design and registry effort across many different participants and allowing the participants to retain some level of ownership of the standardization effort.

COMBINING
COLLABORATION
AND REGISTRY:
THE XML
COLLABORATOR
SOLUTION

As we've seen, collaboration and registry share many of the same design goals. The results of a collaboration process (finalized structures or interfaces) are themselves published as work products in a registry. If a single platform manages both the design of the data structures over time as well as the sharing of those data structures through a registry, then the entire lifecycle of those structures is encompassed by that platform. The atomic level of structure management available through a collaboration platform encourages reuse of those structures, leading to registered structures and interfaces that interoperate with other systems as much as possible.

Blue Oxide has created its XML Collaborator product to serve as both a collaboration platform for the design of information structures as well as a registry for the sharing of those structures with potential users. As we will see,

the XML Collaborator product has been designed from the ground up to be as flexible as possible, and to promote metadata reuse as strongly as possible.

XML Collaborator architecture

The XML Collaborator platform consists of a core metadata tracking database and a series of XML Web service interfaces to that information. The engine also provides a set of shared registry services that allow XML Collaborator servers registered with one another to share information and reuse components across the Distributed Registry Bus.

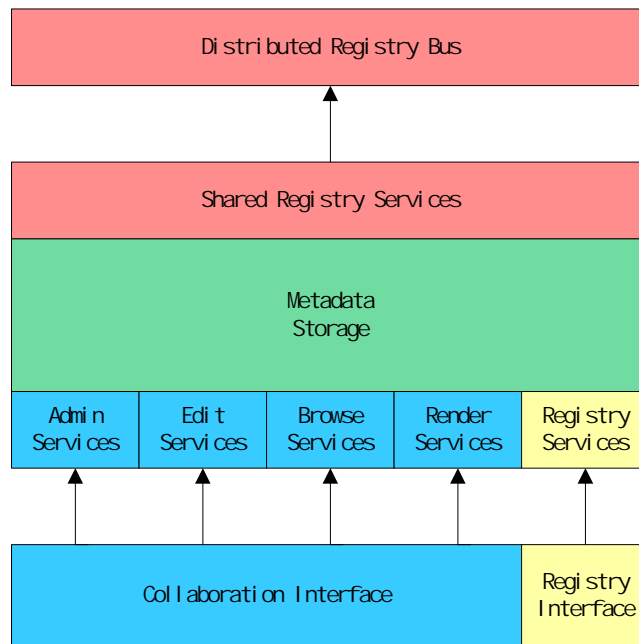


Figure 6: XML Collaborator architecture

The services provided to the user interface include:

- *Administrative services:* These services allow administrative tasks to be performed, such as adding users, groups, and roles, assigning permissions, and so on.
- *Edit services:* These services allow specific structures within the database to be created or modified by users with the appropriate permissions.
- *Browse services:* These services allow users to search the database across the entire XML Collaborator network to discover specific data elements, structures, or interfaces that have particular metadata associated with them.
- *Render services:* These services allow the creation of specific outputs from the structures stored in the database. These include XML

Schemas, DTDs, HTML documentation, database creation scripts, and even parser code for various platforms that can parse and create documents based on the defined XML structures.

- *Registry services:* These services allow the structures and interfaces that have been promoted to the release level by the stakeholders in the design process to be exposed to a larger audience. Again, these structures can be used to create various outputs, depending on the needs of the organization wishing to use the structures in the registry.

XML Collaborator ships with a browser-based client that uses the Web service interfaces to interact with the registry. However, the service descriptions for the engine Web services are published—so other systems could be built that take advantage of the XML Collaborator engine infrastructure but that use some other form of user interface. It will of course also be possible for programmers to interact with an XML Collaborator engine directly through the Web services, allowing more sophisticated behavior such as interfaceless interaction with the repository to be easily implemented.

XML Collaborator features

XML Collaborator 1.0 includes a robust feature set that makes it ideally suited to be the collaboration and registry tool of choice.

Collaboration

- *Real-time collaboration:* Changes made to the metadata repository are reflected immediately through the engine interfaces—allowing teams to easily work together from across town or across the country.
- *Issue and resolution tracking:* Each information component in XML Collaborator may have issues assigned to it by users with the appropriate permissions to do so. The system will not allow a release of an information structure that involves an information component with outstanding unresolved issues.
- *Scratchpad support:* Each user and group has a scratchpad area, where information items can be imported, examined, and manipulated. As a user or group finalizes the design of an information item, it can then be promoted to a position in the namespace hierarchy; from there, it may be integrated into other information structures or released to the registry.
- *Threaded collaboration at every level:* In XML Collaborator, all users, groups, data elements, data structures, enumerations, datatypes, services, and namespaces have their own threaded forum. This allows users to easily communicate specific issues about any aspect of the information components in the system and collaborate on the resolution of outstanding issues.

- *Full-featured security model:* The security model implemented by the platform allows specific permissions to be granted or denied on every possible action in the system. A flexible role system is also provided, where users can be assigned to specific roles for specific branches of the information hierarchy such as reviewer, contributor, and manager. New roles may also be created by system administrators as necessary.

Flexibility and ease of use

- *Information design independent of serialization:* XML Collaborator abstracts the modeling of data structures and interfaces away from any particular implementation. This means that the same system can be used to easily generate XML Schemas, DTDs, HTML documentation, or even scripts for the creation of relational databases based on the defined structures. This loosely-coupled approach to information modeling gives the information model longer-lasting value and makes it much easier to repurpose the information model as necessary.
- *Robust import/export functionality:* XML Collaborator can easily import information metadata from, and export information metadata to, a variety of formats. Existing XML Schemas, DTDs, relational database DDL, and even Excel spreadsheets of data points can be loaded into the system for incorporation into information models. Additionally, the platform can produce all of these as exports of selected information models. The platform can also easily create documentation for a particular information model, as well as parser code in a variety of languages such as Java and C#.
- *Peer-to-peer distributed development environment:* XML Collaborator servers may share information components with one another. The Web service interfaces allow cross-server searches to be easily performed, and information composites may be synthesized from smaller components distributed across multiple servers. If a participating server goes offline for some reason, the remainder of the server network will continue to function normally—if a server needs to call on the missing system for some reason (such as to retrieve information about a particular data element stored there) the Web service will indicate that the particular server is currently offline.
- *Browser-based access to collaboration platform and registry content:* XML Collaborator ships with a client that runs in recent versions of Netscape/Mozilla and Internet Explorer. This enables users with the appropriate credentials to access XML Collaborator without installing a client application, as well as allowing access from any system with a compatible browser.
- *Published API:* The XML Collaborator engine interface is published as a set of Web service calls. This makes it easy for programmers to integrate the XML Collaborator engine's functionality into their own software development efforts, and makes it possible for future user interfaces to be easily implemented.

Management of the design process

- *Atomic versioning:* Within XML Collaborator, each information component that makes up an information structure—data elements (values), data structures, enumerations, datatypes, and services—carries its own version information. As each of these information components is modified over its lifetime, changes are tracked along with the user who made the changes and a short description of the change made. This makes it easy to track who has changed each component of an information model and when.
- *Version tagging:* When an information model is deemed finalized and is ready to be released to the registry side of XML Collaborator, the root data structure in the model is tagged for release. This automatically locks the version for all of the descendants of that composite (both data elements and composite structures) for that release, ensuring that all derivations of that data model (XML Schema, DTDs, etc.) will remain static regardless of other changes to the information components that make up that data model after the release is generated.
- *Namespace hierarchy:* The XML Collaborator platform allows the information components stored in the system to be partitioned into a tree structure of nested subgroups—essentially, allowing the information to be grouped for easier manipulation through the user interface (and to control the security at a more granular level). This allows a sophisticated design process with hundreds of participants to be efficiently managed. Each branch of the information hierarchy is assigned a namespace URI. If the information model is exported to a form that understands namespaces (such as XML Schema), the namespace will be reflected in the created structure.
- *Fine control of data points:* The XML Collaborator platform permits the allowable values for data points to be constrained. This includes a comprehensive enumeration management interface, as well as the typical range and precision constraints.
- *Description fields available at every level:* Descriptive information may be provided for each information component in the system. This information may optionally be included as annotations in exported XML Schemas or comments in DTDs, and is used to build HTML documentation for the information structures.

Registry

- *Robust search mechanisms:* XML Collaborator features a robust set of search services, allowing information components with particular metadata to easily be located across an entire server network.
- *Support for existing standards:* XML Collaborator is designed to support all existing XML, Web service, and registry standards. The 1.0 version of Collaborator includes support for ISO 11179-3, as well as implicit support for ISO 11179-4 and ISO 11179-5. It also provides Web service technical

registration as detailed in the UDDI specifications, and support for the registry functions defined in the ebXML registry specification.

- *Registration of supporting materials:* In addition to XML content, XML Collaborator allows other materials such as UML diagrams of business processes, stylesheets and supporting documentation to be registered within a particular namespace.

Planned enhancements in future releases

- *Support for emergent technologies:* As technologies such as topic maps, RDF, OWL, XLink, XPointer, XML Signature, XML Encryption, and so on mature and become part of the information management process, support will be added to the platform for these technologies. Because the platform is designed with a loosely coupled architecture and a highly abstracted data model, these technologies can be easily added to the platform as they become available.
- *Full ontology support:* Future versions of XML Collaborator will include a full ontology classification system conformant with both ebXML and ISO 11179-2. We are also creating additional interfaces to the system that will be fully compliant with ebXML, ISO 11179, and UDDI, so that other systems that understand these interfaces will be able to access XML Collaborator's registry.

CONCLUSION

In this white paper, we've seen how companies and organizations are facing a significant information design task. The stakeholders in the design process often work for many different companies and organizations, in different locations, each with their own requirements for a shared design. There is a need for a single system that will handle every step of the information design lifecycle—from the identification of individual data elements through to the ultimate sharing of the designed structures and interfaces with interested trading partners. XML Collaborator is the only product that provides this robust, end-to-end solution today.

For more information about XML Collaborator, please contact us at info@blueoxide.com.

© Copyright 2002 Blue Oxide Technologies, LLC.
All Rights Reserved.

Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws. The information contained in this document is subject to change without notice. Blue Oxide makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Blue Oxide shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material. Technical information in this document is subject to change without notice.

Blue Oxide and XML Collaborator are trademarks of Blue Oxide Technologies, LLC in the United States and other countries. All other brand names are trademarks of their respective owners.

Blue Oxide Technologies, LLC.
RR 3 Box 227N
Charles Town, WV 25414
(304) 724-6767

www.blueoxide.com
info@blueoxide.com

