HELSINKI UNIVERSITY OF TECHNOLOGY 24.11.1999
Telecommunications Software and Multimedia Laboratory
Tik-111.590 Research seminar on digital media
Fall 2000: XML

XML Repositories

Kari Kellokoski

46772r

XML Repositories

Kari Kellokoski
Student at HUT Department of Electrical and Communications Engineering

Kari.kellokoski@iki.fi

Abstract

This paper briefly discusses the concept of XML Repositories. As the concept is not only a simple database, the XML Repository is discussed after the requirements of XML databases are considered. Finally couple of examples of available applicationse are discussed briefly.

1 INTRODUCTION AND OVERVIEW

There is lot of expectations on XML, how it should tie up internet, ease the work of application designers and give the control for users to locate and use the information they seek. The technology is present, however, XML is not a solution. It is merely a tool that enables us to take advantage of the enormous information that is –and is not – available.

To that all information to be available, what is needed - of course - is a database. Even XML provides many of the things found in databases: storage (the XML document), schemas (DTDS, XML schema languages), query languages (XQL, XML-QL, QUILT, etc.), programming interfaces (SAX, DOM), it lacks many of the things found in real databases: efficient storage, indexes, security, transactions and data integrity, multi-user access, triggers, queries across multiple documents, and so on. Although an XML document contains data, without any additional software to help process that data, it is no more a database than any other text file.

This paper discusses the tecnologies needed to make data stored in XML documents available through a repository. Repository can be taken as a Database of storing XML data or a Repository defingn data formats. Databases storing the XML data is the starting point for this paper, but that it will go further. XML provides a standard for defining data formats for transportation across the web and this is where the XML Repositories step in. XML Repositories are discussed after the basics - the databases – are discussed, starting from the relation to EDI and contiuning to it’s role in XML schemantic itself.

Goal of this paper is to give a starting point in information of XML storage handling and Repositories; what are the specifications for xml databases, what is a repository and how does it work. Also examples of current tools are briefly discussed.

2 DATABASE FOR XML DATA

2.1 database/Repository Requirements

XML enables many new technologies, including *Internet Search Engines*, *Electronic Commerce, EDI*, *Self-describing BLOBS and Distributed Object file Systems, Data Re-purposing, Content Personalization (intelligent pull, agent accumulation, and push), Customized Bandwidth Allocation, Individual Content Cache,* etc. All of the exciting applications described above require data persistence, and these requirements are very different from the requirements of monolithic file storage we are used to. XML extends HTML's simple unidirectional linking, adding support for links to multiple targets, indirect addressing and bidirectionality. Handling this rich linking requires a storage mechanism with far more powerful management of references between objects than that provided by the file system or relational databases. In order to effectively address the XML opportunity the storage mechanisms must also be able to understand the structure of XML content—which is composed of a dynamic number of objects—while scaling effectively to handle increased usage load and data volume. Furthermore, XML's object centric focus will create the need for an API that enables rich object-centric manipulation from object-oriented languages like C++ and Java. In other words, what is needed is an XML-aware object repository. Only an object database management system (ODBMS) can maintain information about XML document structure in a scalable manner while handling standard data types and BLOBs with rich hyperlinking and navigation in the database.

XML documents fall into two broad categories: *data-centric* and *document-centric*. Data-centric documents are those where XML is used as a data transport. They include sales orders, patient records, and scientific data. Their physical structure -- the order of sibling elements, whether data is stored in attributes or PCDATA-only elements, whether entities are used -- is often unimportant. A special case of data-centric documents is dynamic Web pages, such as online catalogs and address lists, which are constructed from known, regular sets of data. Document-centric documents are those in which XML is used for its SGML-like capabilities, such as in user's manuals, static Web pages, and marketing brochures. They are characterized by irregular structure and mixed content and their physical structure is important.

To store and retrieve the data in data-centric documents, you will need a database that is tuned for data storage, such as a relational or object-oriented database, and some sort of data transfer software. This may be built in to the database or might be third-party middleware. Depending on your needs, you may need Web publishing abilities as well.

To store and retrieve document-centric documents, you will need a *content management system*, which is a system designed to store content fragments, such as procedures, chapters, and glossary entries, as well as document metadata, such as author names, revision dates, and document numbers. Such systems generally include editors, version control, and the ability to construct new documents from existing fragments. Although content management systems use a database for storage, this is hidden from the user.

[16,18]

2.2 Comparing Database Types

2.2.1 File System Storage of XML Data

HTML storage management is almost always implemented using flat file storage. This is because HTML, lacking any definable structure, is stored as a monolithic block. Using the file system this way provides acceptable functionality, and therefore wins out because it is extremely easy to implement. There are a number of tools that build on the file systems functionality, and file systems are included in operating systems at no additional charge. XML, however, has very different storage requirements. XML applications must store and index the fine-grained elements as well as the document structure. In addition, they must be capable of linking these fine-grained elements directly to each other and to a variety of data types containing associated information. The increased demands implied by this functionality mean that additional care must be taken to build a system which scales under increasing load. Attempts to parse XML data of any complexity would overwhelm the capabilities of the file system to maintain the rich linking structure and semantics of the data. Of course, the alternative is to store the XML data as BLOBs and then parse it on the fly each time it is used. This results in sub-optimal performance, because of repeated parsing. In addition, once the data were parsed, the file system could not execute the complex data manipulation required. In addition, this BLOB storage approach undermines the ability to link various disparate elements into a rich tapestry of information that models real-world usage cases. Of course, implementing all of these features on top of the file system is a possibility, through custom development, but this would essentially recreate object database functionality from scratch.

2.2.2 Relational Database Storage of XML Data

Relational database management systems (RDBMS) are the other plausible candidate. Unfortunately, their table-based data model is very poorly suited to the hierarchical, interconnected nature of structured XML content. Never the best systems for managing variable length data and BLOBs, RDBMSs are further hampered by the fact that they must represent the tree structure of XML content with an inefficient set of tables and joins. Relational databases disassemble the XML objects in order to fit them into their tabular architecture. As a result the XML object's structure and semantics are either lost, minimizing its value, or they must be duplicated in the design of the database. Duplicating the structure and semantics of complex XML objects in the design of the database is very difficult, particularly if the structure of the XML data is variable, as it almost always is. The rigidity of the relational design is a poor fit with the dynamic assembly and manipulation of XML data. Relational databases also cannot handle object-level locking, the best they can provide is row-level locking. Since relational databases decompose XML elements into various tables, linked via keys, it is very difficult to implement an effective locking scheme that doesn't dramatically hinder concurrent use and scalability. In concurrent editing environments, there will be an increase in demand for related objects from disparate users. Relational databases respond by locking entire rows across multiple tables. This can cause unacceptable performance degradation if multiple users are requesting different objects that are locked via this broad locking scheme. If the DBA responds by separating the information into a larger number of more granular tables, the performance is degraded by the number of joins required to model the richly linked structure of structured XML content. In addition, relational databases are typically too heavyweight to form an infrastructure for embeddable storage and require substantial development to adapt to the complex structure of structured XML content. Quite simply, relational databases, while excellent for many purposes, are not architecturally compatible with the storage needs of XML data.

2.2.3 Object Database Storage of XML Data

The architecture of object databases is ideally suited to handling XML data, in fact the adoption of XML data could be the "killer application" for the object database market. Object databases are designed to handle objects in their native forms. The objects then maintain their own data, methods, relationships and the semantics of the whole model. This is ideal for the creation and management of hierarchical XML trees, while providing both hierarchical tree navigation and rich link traversal. For example, traversing to the other side of a tree in the two dimensional relational model forces the developer to climb up the tree, through joins, and then back down the other side. The rich relationship linking of an object oriented model enables both hierarchical navigation and rapid branch traversal, reducing computation and increasing performance. Object databases are also designed to handle arbitrary, variable-length data types and interrelated data. This is critical due to the various data types linked within structured XML content. XML also enables an ever-changing web of relationships between hyperlinked data elements, such as on-the-fly creation of documents. Object databases, with their flexibility and rich relationship management are ideal for managing this type of information.

Those object databases that allow object-level locking, provide for a much more granular locking than relational or file system-based solutions. This granular locking is critical for user scalability, since it limits the conflicts between user requests for data. Object databases are also designed to handle larger than memory content, providing for content scalability. Object databases also offer more simplified creation and management of distributed partitions, which further addresses the issue of database volume scalability and distributed implementation. In short, object databases were designed to address the very requirements that XML is just now starting to force upon tomorrow's storage solutions.

Just as most software developers would never consider developing their own encryption technology for a new product, licensing it instead from a third-party vendor selling best-of-breed solutions, it makes sense for XML developers to seek a third-party storage engine that instantly provides them with the feature set they require for effective XML data management. These applications need a mature database engine well-suited to the structure and data types associated with XML.

2.2.4 The Ideal Repository for XML Data

The ideal XML repository should address the needs of XML as well as the needs of the associated applications. In evaluating the requirements of applications in this field, the following criteria are critical: (1) scalability, (2) language support, (3) ease of programming and (4) embeddability. Scalability will be very important since the XML applications described above will run on both the client and the server. It is important that the object database scale down as well as up, while leveraging the same APIs, to simplify application development. Language support is also important. Ease of programming is critical due to the compressed development cycles, particularly in the Internet. Embeddability encompasses two criteria--zero-management and low memory footprint. Embeddability is an interesting issue since it has both short and long-term ramifications. In the short-term, embeddability it important because most initial XML applications, lacking sufficient XML support in the file system, will build-in this support. However, long-term the object database will replace the standard file manager running on top of the file system. This of course will make embeddability an absolute requirement.

Storing and retrieving XML data is only the start. In addition to standard XML data storage, the ideal repository would offer tightly integrated XML-specific tree navigation, versioning, management of arbitrary links, import/export, publishing of structured content on the web, support for object-oriented programming languages as well as common scripting languages, and more. Building these facilities on top of a object database are non-trivial, yet they are critical to the actual process of managing and manipulating the XML data stored in the object database.

[16]

2.3 Conclusion

Because XML applications have a far wider scope than their HTML counterparts, in terms of both application domain and type of data being managed, existing solutions for HTML storage are not sufficient for XML. File system-based solutions do not support XML’s rich linking and hierarchical tree structure, negating XML’s value-add. Relational databases are based on a two-dimensional table-based architecture that is also ill-suited to the needs of XML. Attempts to force a relational database into storing XML will result in sub-optimal performance, concurrent access and scalability due to the architectural mismatch between XML content and relational databases. The only database architecture that is suited to the demands of XML is the object database. Only object databases support granular element access and locking for superior concurrent access, rich high-performance hyperlinking and fast hierarchical navigation.

But there is an other point of view; Specialized XML servers and object databases being the natural storage point for XML documents and data, there is a question that will arise eventually: "So where does all this XML data come from?" The answer, of course, is: "From the relational databases that underpin the majority of business applications in use today." For most XML applications, the creation and ongoing maintenance of a separate "XML repository" is an unnecessary development expense. Instead, all most companies need to do is take information from existing databases and render it as XML so it may be shared and consumed more easily. Of course, if the relational database in question has object capabilities, such as Oracle8i, then the task of rendering and updating data as XML is greatly simplified.

[16,17]

3 REPOSITORIES

3.1 The Role of Edi

In the growing world of electronic commerce, common standards are needed, especially in an environment in which machines talk to machines. XML is a standard that governs how data is to be represented. It has provisions for defining data elements plus extending and customizing data structures. The power of XML is its flexibility to define an arbitrary data structure for both machines and people. A data structure defined in XML can be validated and easily parsed by using readily accessible, standard parsers. Its standards are predominantly applied to data schemas. What ASCII is to character encoding, XML is to data structures. In summary, XML gives you the power to specify the vocabulary but it doesn't impose or attempt to standardize the individual elements of the vocabulary.

XML isn't a dictionary, only a means of formulating one. What is required is a standard that can embody a business dictionary in the context of business processes. This is where EDI comes in. With its long history, EDI provides common meaning to business transactions through the use of standard business documents. These documents - purchase orders, invoices, etc. - represent the contents of day-to-day business transactions. The set of business documents encapsulates business events and processes.

There are many standard bodies associated with EDI. The common ones are EDIFACT (EDI For Administration, Commerce and Transport), used primarily in international circles, and ANSI X.12, widely used in North America. EDI embodies more than just data definition, as it also dictates the transporting and administering of the documents.

While the EDI documents provide a rich platform for business communication across different industries, from retail to government, they're sometimes viewed as being too rigid for certain businesses. Although the EDI documents provide optional capabilities, the documents can't be easily extended like XML. As business processes change, the demand for new fields arises. Users of EDI begin to tweak the standards by placing information where it doesn't belong in the documents. Data that didn't fit often ends up in unused fields. The semantic meaning of the fields then deteriorates, because now it represents information other than its original specification. This ad hoc solution works as long as all the trading partners are aware of the tweaks.

The concept of EDI is a good one. However, its implementation of a rigid document set sent over a proprietary network infrastructure - and expensive translator software for back-office integration - tends to inhibit the participation of new players in the world of electronic commerce. The XML technology opens the door for companies that didn't have the opportunity to participate in EDI in the past.

[3,13,14]

3.2 XML and Industry Standards

Many major forces, such as RosettaNet, CommerceNet and OAG (Open Applications Group), are alsotrying to leverage the extensibility nature of XML to fill the grammar and vocabulary gap. These are organizations with big players that are undertaking the daunting task of specifying the business dictionary using XML. Within these organizations you'll find EDI standard bodies such as ANSI ASC X.12 and UCC (Uniform Code Council). New XML schemas are being created every day, which creates a new problem. Which definition should I use? Which dialect should I speak?

To tackle part of this problem, organizations such as Microsoft's Biztalk.org and OASIS's XML.org are taking shape. Their purpose is to act as a clearinghouse for XML documents by soliciting business partners to collectively create a vocabulary repository. Different XML schemas are published in these clearinghouses for others to consume and use. They represent a hub for the business community of business dictionaries specified in XML.

The EDI experience has taught the electronic-commerce community about the need to customize and extend. An order document in the auto industry will be different from one in the electronic industry. The health-care industry will have business processes substantially different from processes in the retail industry. Each vertical industry will demand its own set of business documents. A horizontal process, such as order entry, will be different for each industry, resulting in different business documents to be transacted. The combination of the expressive power of XML and the development of big organizations to produce common business dictionaries is finally beginning to bridge the gap of our communication stack.

[14]

3.3 XML Repositories and Registers

XML provides a standard for defining data formats for transportation across the Web. XML data format types are expressed in the form of XML schemas. An XML schema is a document that describes a set of XML document instances. In that sense it's like an XML document template. The only way that enterprises will agree on common schemas is if there's a shared resource that makes the same schema available to multiple organizations. Such resources should be governed by industry consortia so that multiple organizations can be represented and the acceptance criteria can be as unbiased as possible.

However, shared resources need careful management. As different industry verticals define unique schemas for exchanging XML-based information, the proprietary nature of these schemas will lead to a lack of portability across different e-business environments. There will also be an explosion in the number of redundant schemas that will emerge to express the same type of data. There is a growing need in e-businesses for compatible processes and vocabularies to reduce this redundancy and the consequent complexity. For industries to exchange data using XML across multiple enterprises, standard repositories are needed for sharing vocabularies. These repositories serve as data stores for DTDs and schemas, XML-based directory mechanisms, database structures, UML modeling tools, glossaries for relationships, context-specific terms and so on. These repositories are usually owned by consortia of industry verticals that hash out things like the meaning of an "SKU" or the elements of an "invoice." Repositories will eventually contain standardized business components, tags, and industry terms and definitions.

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

Organizations access XML repositories through an open application program interface or API. This API will allow automated or manual queries from organizations needing to exchange data with trading partners. Organizations download the package of XML-enabled objects (document-type definitions, tag sets, style sheets, processing routines, product identifiers, etc.) needed to do business with others in this industry. Organizations can then exchange XML data that feed directly into their business systems and those of their trading partners. Repositories can also provide a temporary holding bin for data used in collaborative problem solving

Figure below shows a schematic of the role of repositories in the context of XML/EDI - business information exchanges using XML as the syntax.

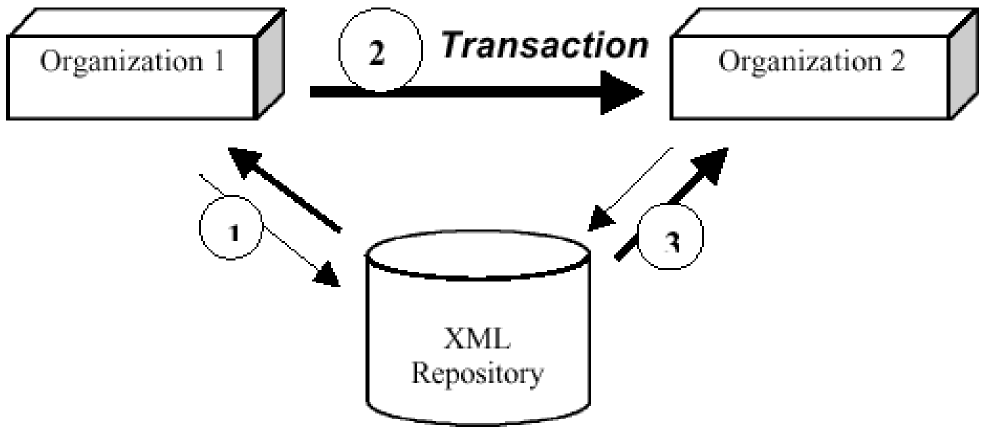

Figure 1: XML/EDI transaction

In Step 1, a user or software process in Organization 1 queries the global repository for those common business objects to be passed to trading partners such as Organization 2. In step 2, references to the identified business objects are exchanged as part of the set up process for the transaction by the trading partners. In step 3, the references are used to map received data into the organization's application system. The intent of the global repository is to be a dynamic mechanism that responds through an open API.

[1,3,15]

3.4 Repositories' role

XML repositories provide an online source to acquire the appropriate tag templates, document-type definitions (DTDs), database schema, software code or routines, and other objects needed to conduct electronic commerce using XML. As a result, organizations, particular smaller enterprises, can quickly develop or expand their ability to conduct electronic commerce.

EDI requires trading partners to map their business data to the formats of the transmission standards, an often difficult and time-consuming process. As mentioned before, EDI often requires using expensive translator software and networks that charge according to transmission volumes. XML repositories, on the other hand, provide the means to exchange business data with a much smaller financial investment by:

- Extending the reach of electronic business well beyond the simple exchange of electronic documents to include the mapping and processing of data directly into the organizations business systems
- Covering a wider range of interactions among trading partners, such as real-time exchanges, that often do not fall cleanly into the normal definition of business transactions
- Using the Internet, with appropriate security, which offers a more economical means of transmitting data than traditional value-added networks
- Taking advantage of the next generation of XML-enabled applications software that can interact directly with the transmitted data from trading partners, circumventing the difficult mapping process

Applying the capabilities provided by such an XML repository makes it possible to create and deploy business applications that can support the needs of many thousands of trading partners. This 'self service' approach, allowing many users to share the development efforts, is the key to scalability compared to traditional EDI implementations.

The single biggest role for XML repositories is taming semantic dispersal brought about by the explosion in electronic information systems over the last 25 years. Early EDI systems avoided this problem simply because the number and range of installed systems was relatively low. At this stage the use of fixed structure transactions that trading partners had to adhere to limited the scope for dispersion. As the numbers of parties involved in EDI increased efforts were made to combine the existing message structures into a set of shared semantics, at first at national level (e.g. the ANSI X12 effort) and then internationally (e.g. the UN/EDIFACT work). Compromises had to be made as part of this coming together, with the result that many of the components of today's EDI messages are overloaded with conflicting sets of semantics.

[1,3,13]

4 AVAILABLE SOFTWARE TOOLS

The number of products for using XML with databases is growing with amazing speed -- new products seem to enter the market weekly. It is obvious that XML is approved well among developers and there is a need for XML database applications and repositories, but what kind of applications are present and used nowadays? Answer for that question is tried to met by taking a glance of two products below; one XML Database product and one XML Repository, Oracles 'heavy duty' database Oracle 8i and Microsoft's BizTalk framework.

4.1 Oracle8i Database

If the key requirement is to be able to efficiently read and write XML to and from the database, developers should look hard at Oracle8i. Designed from the ground up to be the database for the Internet, it includes native support for Internet standards such as Java and XML. In fact, at the heart of the Oracle8i database is a highly scalable Java engine tuned for server-side Java application development, giving Java the ability to scale to thousands of heavy-duty concurrent connections. In addition, Oracle's XML capabilities are so much a part of the database that Oracle8i can run its XML components for Java completely within the database to deliver the scalability required by today's Internet applications.

Oracle 8i can store XML documents in three different ways: in the Internet File System (iFS), using the XML SQL Utility for Java, and as a BLOB that can be searched using the Oracle Intermedia XML Search. Oracle 8i also includes a number of other XML-related tools, the most interesting of which is the XML Class Generator, which Can generate Java classes from a DTD.

With iFS, one or more type definition (mapping) files defines how to map an XML document as a tree of data-specific objects. iFS uses these mapping files both to construct tables in which the XML document can be stored, and to transfer data between XML documents and the database. Of interest, iFS can return data from the database directly as objects instead of XML documents, which is useful to many applications. iFS also supports content management features such as check-in/check-out and versioning.

The XML SQL Utility for Java used with Oracle 8i, it models XML documents as a tree of data-specific objects, since Oracle 8i supports SQL 3 object views.

Oracle Intermedia XML Search is a utility that can, "Automatically index and search XML Documents and Document Fragments of any size up to 4 Gigabytes each. [It has] powerful XML document searching including hierarchical element containership, doctype discrimination, and searching on XML attributes."

Oracle 9i (announced 2 Oct, 2000) includes "Native XML Database Support (XDB)", which introduces a new object data type (XMLType) and "features ... 'navigational' access and search for XML documents."

4.1.1 Additional XML features in oracle 8i

- XML-Enabled Object Views Over Relational Data': Object Views allow information in relational tables to be materialized on the fly as a richly-structured XML views of any information in the database. ... XML information can be easily inserted into the database through the Object Views.
- The XML SQL Utility (XSU) provides the user with the following functionality:
  - Generate an XML document (string or DOM) given a SQL query or a JDBC *ResultSet* object.
  - Extract the data from an XML document, then insert the data into a DB table, update a DB table, or delete corresponding data from a DB table.
- Oracle Intermedia XML Search': Automatically index and search XML Documents and Document Fragments of any size up to 4 Gigabytes each. Powerful XML document searching including hierarchical element containership, doctype discrimination, and searching on XML attributes.
- Oracle8 JServer Java Virtual Machine': ... this fully-compliant Java 1.2 virtual machine shares the memory address space of the data server. This allows mission-critical, data-intensive Java and XML-processing code to run with in-memory data access speeds using standard JDBC interfaces. ...
- Oracle XML Developer's Kit': The Oracle XDK contains the basic building blocks for reading, manipulating, transforming, and viewing XML documents. ... The Oracle XDK consists of the following items: XML Parsers, XSL Transformation Engine, XML Class Generator, XML Transviewer Java Beans, XML SQL Utilities, XSQL Page Processor and Servlet, JDeveloper, and Business Components for Java."

[6,7,8,9,10,11,18]

4.2 BizTalk Schema Repository

BizTalk has been developed by Microsoft and is supported by many organizations. These are technology vendors such as SAP, through to technology users like Boeing. BizTalk is not a standards body. Instead, it is a community of standards users, with the goal of driving the rapid, consistent adoption of XML to enable electronic commerce and application integration.

Microsoft has taken 'the race of the repositories' seriously. It has to offer three concepts to XML application designers; *the BizTalk repository*, *the BizTalk Schema framework* and *the BizTalk Server*. Microsoft will natively support the BizTalk Framework in its product line and will publish XML schemas to the BizTalk Framework Web site for public use. Other software vendors supporting the BizTalk Framework have also made this commitment.

Microsoft has claimed that more than 150 member organizations participate in BizTalk and more than 500 schemas have been published to BizTalk.org.

4.2.1 The BizTalk Framework

The BizTalk Framework is an Extensible Markup Language (XML) framework for application integration and electronic commerce. It includes a design framework for implementing an XML schema and a set of XML tags used in messages sent between applications. Microsoft Corp., other software companies and industry standards bodies will use the BizTalk Framework to produce XML schemas in a consistent manner.

The BizTalk Framework itself is not a standard. XML is the standard. The goal of the BizTalk Framework is to accelerate the rapid adoption of XML. BizTalk Framework schemas — business documents and messages expressed in XML — will be registered and stored on the BizTalk.Org Web site. Any individual or organization can download the framework and use it to implement and submit XML schemas to the Web site. As long as the schemas pass a verification test, they are valid BizTalk Framework schemas. The BizTalk.Org Web site will provide an automated submission and validation process. Individuals or organizations can freely use XML schemas from the BizTalk.Org Web site within their applications, as long as the schema is published for public use.

Businesses will also have the option of publishing their schemas on the BizTalk.Org Web site in a secure area for private use between trading partners. A steering committee composed of software companies, end users and industry standards bodies will provide guidance on how the BizTalk.Org Web site is organized and managed.

[4,5,12]

5 CONCLUSION

While XML has evolved from SGML and HTML, its impact will not be evolutionaryit will be revolutionary! XML will transform the Internet from a massive collection of unmanageable data into the intelligent transport we have all been waiting for. The development community is just beginning to recognize the far-reaching benefits of XML as a standard, flexible, structured content format. Existing XML DTDs such as CDF, for push channel management, and OSD, for on-line software distribution, merely hint at the types of applications that will benefit from XML’s unique characteristics.

While XML’s impact is revolutionary the thing where we should be looking at is how to make use of it. XML Repositories together with suitable, efficient and reliable databases, are the way of making that happen. Repositories are the solution for efficient handling, creating and maintaining of business standards and that seems to be the direction companies are heading. But it is not happening fast. The biggest slowing factor for the growth of this XML–based information ’society’ is the current databases containing the data along with current procedures companies are having in their business’s. Questions relevant are: will companies give effort to change their systems? What will they benefit if they do?

In the age of the Internet, XML offers an opportunity to rethink the implementation of EDI but it can't replace it by itself. The standard business dictionaries stemming from standard organizations are crucial to the success of business-to-business communications. The extensibility of XML will ease the traditional EDI pain of being too rigid in definition. The pace of competition is extreme and this means businesses need to be as agile as ever. This competitive environment forces businesses to change and upgrade their processes frequently. These changes in processes may lead to changes in messages. How well will XML, along with the armies of standard bodies, keep pace? Only time will tell.

Traditional EDI is now being by-passed by events and the Internet. However the Internet, which has no inbuilt semantics other than those required for the presentation of data, has in turn become overloaded with information. XML is seen as the solution to this problem by allowing semantics to be associated with data being transmitted over the Internet. XML by itself is merely a syntax and a tool. It will not of itself allow the adoption of standardized sets of semantics. XML Repositories are there to tame the semantic dispersal.

REFERENCES

[1] White Paper on Global XML Repositories for XML/EDI (The XML/EDI Group Feb, 1999)
http://www.xmledi-group.org/xmledigroup/repository/RepWPv1.PDF
Ref. 25.10.2000

[2] The Role of Document Type Definitions in Electronic Data Interchange (Martin Bryan, The SGML Centre)
http://www.sgml.u-net.com/xml-edi/edi-dtds.htm
Ref. 25.10.2000

[3] XML/EDI Repositories, Q&A
http://www.xmledi-group.org/xmledigroup/repository/Rep-Q&A.htm
Ref. 25.10.2000

[4] BizTalk Framework 2.0 Draft: Document and Message Specification (Microsoft Corporation, June 2000)
http://msdn.microsoft.com/xml/articles/biztalk/biztalkfwv2draft.asp
Ref. 23.11.2000

[5] BizTalk Framework
http://www.biztalk.org/Biztalk/framework.asp
Ref. 23.11.2000

[6] Product info & articles (Oracle Corporation, 2000)
http://www.oracle.com/ip/deploy/database/8i/index.html?ee.html
Ref. 23.11.2000

[7] Oracle XML SQL Utility – XSU (Oracle Corporation, 2000)
http://technet.oracle.com/tech/xml/oracle_xsu/
Ref. 23.11.2000

[8] Using XML in Oracle Database Applications (Oracle Corporation, 2000)
http://technet.oracle.com/tech/xml/info/index2.htm?Info&htdocs/otnwp/

about_xml.htm
Ref. 23.11.2000

[9] Using XML in Oracle Database Applications (Oracle Corporation, 2000)
http://technet.oracle.com/tech/xml/info/index2.htm?Info&htdocs/otnwp/
about_oracle_xml_products.htm
Ref. 23.11.2000

[10] Using XML and Relational Databases for Internet Applications (Steve Muench, Consulting Product Manager & XML Evangelist, Oracle Corporation)
http://technet.oracle.com/tech/xml/info/htdocs/relational/index.htm#ID795
Ref. 23.11.2000

[11] XML Support in Oracle8*i* and Beyond (an Oracle Technical Whitepaper November 9, 1998, Oracle Corporation)
http://technet.oracle.com/tech/xml/info/index2.htm?Info&htdocs/xml_twp.html
Ref. 23.11.2000

-Schema repositories

[12] Schema Repositories What's at stake (Liora Alschuler, Jan. 26, 2000)
http://www.xml.com/pub/2000/01/26/feature/index.html
Ref. 25.10.2000

-XML/EDI

[13] Guidelines for using XML for Electronic Data Interchange, Version 0.05 (25th January, 1998, Editor: Martin Bryan, The SGML Centre)
http://www.geocities.com/WallStreet/Floor/5815/guide.htm
Ref. 25.10.2000

[14] XML: The Next Generation EDI? (Kang Lu, SYS-CON Publications, Inc 2000)
http://www.sys-con.com/java/xml/lu/
Ref. 25.10.2000, Registration required

[15] Registering XML (Ajit Sagar, SYS-CON Publications, Inc 2000)
http://www.sys-con.com/xml/archives
Ref. 25.10.2000, Registration required

[16] XML: The Foundation for the Future (By Mike Hogan, POET Software,A Sponsor Member of OASIS)
http://www.oasis-open.org/html/xml_foundation_future.html
Ref. 25.10.2000

[17] XML and Databases (Copyright 1999, 2000 by Ronald Bourret September, 1999, Last updated November 2000)
http://www.rpbourret.com/xml/XMLAndDatabases.htm
Ref. 23.11.2000

[18] XML Database Products (Copyright 1999, 2000 by Ronald Bourret September, 1999, Last updated November 2000)

http://www.rpbourret.com/xml/XMLDatabaseProds.htm
Ref. 23.11.2000