Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2002 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2002

THE EXTENSIBLE MARKUP LANGUAGE (XML) AS A MEDIUM FOR DATA EXCHANGE

Meg Murray
Kennesaw State University

THE EXTENSIBLE MARKUP LANGUAGE (XML) AS A MEDIUM FOR DATA EXCHANGE

Meg Murray
Kennesaw State University
mcmurray@kennesaw.edu

Abstract

The amount of information being collected and stored electronically continues to increase as does the need to share this data among disparate applications and non-compatible computer systems. The eXtensible Markup Language (XML) was introduced to meet this challenge by providing a standardized way to exchange data. The adoption of XML is occurring rapidly, and XML is positioned to thrive in the electronic marketplace. A main premise behind Microsoft's .Net strategy and the recent release of Sun's J2EE platform is the belief that XML marks a turning point in the evolution of the Internet and computing architectures. This tutorial includes an exploration of XML and its corresponding components, technical implementation requirements, the development of schemas for defining industry standard data definitions, a scenario employing XML technologies and the potential impact of XML on information systems.

Keywords: XML, eXtensible Markup Language, data exchange, emerging technologies

Introduction

The amount of information being collected and stored electronically continues to increase as does the need to share this data among disparate applications and non-compatible computer systems. New methods are required to handle the storage, retrieval, presentation and exchange of this vast amount of information efficiently and effectively. The eXtensible Markup Language (XML), created in 1996, was designed to address this challenge. A primary objective behind XML was to extend the capabilities of Web technologies to include a standardized way to exchange data and yet be easily implemented with existing systems. XML is poised to be the next Internet standard for computer-to-computer exchange of data (Herman, 2001). Consequently, the implications of XML permeate not only technology issues but also inter-business communications and industry standards of data exchange.

XML is part of larger system known as SGML (Standardized Markup Language). SGML was introduced more then twenty years ago to provide a framework for device independent representation of text in electronic form. The same international standardization body, the World Wide Web Consortium, known as W3C, that oversees the development of SGML also oversees the development of XML and HTML (HyperText Markup Language), the common language of Web pages. The role of the W3C, through its member organizations, is to lead efforts to standardize Web based technologies (World Wide Web Consortium, 2000).

Interest in XML is high, but the newness of the technology and the evolving nature of XML standards means that a learning curve surrounding XML technologies. Even given these shortcomings, XML is fast becoming a transformational technology. XML is being adopted at a rapid rate and being used in a variety of IT development projects. A main premise behind Microsoft's .Net strategy and the recent release of Sun's J2EE platform is the belief that XML marks a turning point in the evolution of the Internet and computing architectures. While most expect that it will take a few years for XML to transcend the technology landscape, it is moving ahead at a rapid pace and will continue to mature into a key technology in information systems.

While the resources associated with XML are many, a word of caution should be noted. XML is a young technology and many facets are still evolving and are not yet standardized. Its status presents many challenges to the XML developer. For example, current browsers do not support XML fully and it will be a while before these technologies converge. Further, supporting

technologies such as XSL (the style sheet language for rendering presentation of XML data) are proposed recommendations since they are still under development and review. Until XML associated technologies, such as XSL, mature to an accepted recommendation by the W3C, they will not be widely implemented. On the other hand, a primary goal of XML is to serve as a foundation for data exchange. XML as a medium for the exchange of data is operational. Document Type Definitions (DTD) specifications were approved with the original XML specification. Schemas, which are quickly becoming the preferred choice for data definitions, received approval in May 2001. XML, itself is stable. The W3C did issue a public working draft of the next version of XML (V1.1) in December 2001. This new draft does not contain major changes to XML v1.0. Instead, it includes specifications for character data included in the new version of Unicode (v3.2) extending the written languages fully supported by XML (World Wide Web Consortium, 2001).

XML is being applied in many areas because XML promises to be a neutral method for exchanging data between two systems or applications (Patrizio, 2001). The top five uses of XML are reported to be for:

- data exchange,
- Web services to exchange data between systems,
- content management,
- Web integration with new kinds of devices such as PDAs and
- Managing computer application configuration data (Wahlin, 2002).

The focus of this tutorial is on XML for data exchange.

XML Structure and Vocabulary

XML is not a product, programming language, or any other type of easily delineated technology. XML is a meta standard that provides a standardized way to describe and define data. Herman (2001) summarized what this means. "By itself XML does not provide any specific data standards. Instead, XML provides a standardized language for creating such standards. That is why XML is called 'extensible;' it will be used by others to develop industry-or function-specific data definitions". XML is used to create 'self-describing documents' or a document that includes information describing what the document contains. XML, being a markup language, the 'describing information' is placed within tags. The advantages of this approach are numerous.

- XML is not limited to a fixed set of element types, or tags. For example, if XML is to be used to tag a document that describes a recipe, tags may be chosen to represent each ingredient as an *item*, or as an *ingredient*, or as *groceries* – the decision is made based on the project and preference of the developer.
- XML documents are simply plain text files. XML documents are easily readable by people, by programming languages and by other applications. In addition, XML files are easily transferred across a network and can be passed using HTTP, the transport protocol of the Internet.
- XML technologies are structured around a collection of documents (files) and other associated resources. The premise behind XML is that it separates data from presentation. With XML, systems can exchange structured data, interpret that data, and display the data in any number of different ways. At the core, is the XML document. An XML document contains tags, elements and corresponding data or content. other files such as schemas and stylesheets are associated with this XML document. These types of files do not contain data per se, but contain definitions or descriptions of how the data in the XML document should be handled. For example, an XML style sheet tells a Web browser exactly how to display the data contained in the XML document while the XML schema contains the definitions of the tags used in the XML document.

Schemas are often referred to as dictionaries or vocabularies that serve as a uniform source for data definitions. They form the foundation for data exchange using XML. The schema specifies a set of rules that defines or constrains the contents of an XML document. Basically, the schema is a coded list that identifies what tags in the XML document describe data and what constraints are put on that data. The power of schemas is that they can be shared by many different XML documents. Several initiatives are underway to develop industry specific schemas to facilitate the exchange of data between different organizations.

The Document Type Definition (DTD) is the predecessor to the schema. The DTD and schema perform the same functions but the schema is expected to replace the DTD. The primary difference between the two is that schemas are written in XML while DTDs have their own syntax. Another distinction between the two is that the schema supports datatypes where the DTD only supports strings. This difference requires a receiving application to convert the data to a different data type, such as a number, if necessary. While DTDs can be used in conjunction with schemas, they are not introduced in this tutorial.

XML is a highly structured markup language meaning it requires its users to follow its rules explicitly. The foremost rule of XML is that it must be well-formed. A well-formed document is one that is properly formatted and follows the basic rules of XML as defined in the W3C XML Specification. One of the primary reasons that the language is extensible is that its rules are rigid. To use XML programs in many different ways, all of the programs must follow the same rules. Other markup languages do not require such standards, and thus tend to be restricted to one initial purpose or project. For example, most HTML documents tend to merge describing the data with describing the layout or format of the data. If these two ideas are interwoven, then the data can only be used for that purpose – it cannot be presented in another layout without editing the code. While the strict requirements of XML are sometimes seen as a disadvantage, well-formedness supports the core XML advantage of extensibility.

An XML parser is used to read an XML document and verify that its contents are well-formed. (Browsers such as Microsoft's Internet Explorer include an XML parser. Several XML parsers have been written. Parsers are included with commercial XML development environments, included with books on XML or can be found on the Internet.) The parser will identify any error found but will not attempt to fix it. Consequently, for XML documents to be usable, they must be well-formed.

An XML document associated with a DTD or schema must also be 'valid' in order to be usable. Validation is the process whereby the XML document is compared against the data specifications defined in its corresponding DTD or schema. The XML document is considered valid if it is well-formed and it meets all the constraints listed in the DTD or schema. XML documents that do not have a corresponding DTD or schema do not need to be checked for validity.

Creating an XML Document

The basic components of a markup language such as XML are tags and elements. Tags are labels delimited by angle brackets while elements refer to tags plus their content. The basic rule of XML is that all elements must be surrounded by a beginning tag and an ending tag. Beginning tags are identified with the '<' and '>' symbols while ending tags are identified with '</' and '>'.

Beginning tag: <TITLE>
Ending tag: </TITLE>
Element: <TITLE> XML Tutorial </TITLE>

An XML document is plain text and may be created in any text editor including Notepad which comes with Windows operating systems. XML documents are identified by a three-character filename extension of .xml.

The basic building blocks of XML are elements and attributes. A complex XML document contains other components as well but this tutorial is limited to the elementary principles of creating a simple XML document. Figure 1 is an example of a simple XML document followed by detailed explanations of the annotations.

Description of the XML Document Components

Prolog

The prolog contains the XML Declaration which is always the first line of any XML file. There should be nothing, including white space, before the declaration. The XML Declaration tells the processor which version of XML to use. A simple declaration may be: <?xml version="1.0">. A complex declaration may include more information, such as: <?xml version="1.0" encoding="UTF-8" standalone="yes"?>. "1.0" describes the version of XML being utilized (currently there is only one version of XML); "UTF-8" describes the language encoding such as English ASCII; and "yes" or "no" indicates whether or not the document relies on markup declarations defined external to the document.

Comment

Comments may be included in XML files. They are ignored by the parser. Comments are included between the tags "<! --" and "-->".

Root Element with Schema

The most important component of the XML document is the element. All XML documents must have at least one element. The first element is known as the root element (may also be referred to as the document element) under which all other elements are nested. In Figure 1, the root element is 'Order_request.' The attribute to the root element points to the schema by which this document is validated. The schema may be located anywhere so long as it is accessible.

Element with Content

The most common format for elements is: start-tag (), content and end-tag (). This format is used for the majority of elements in Figure 1.

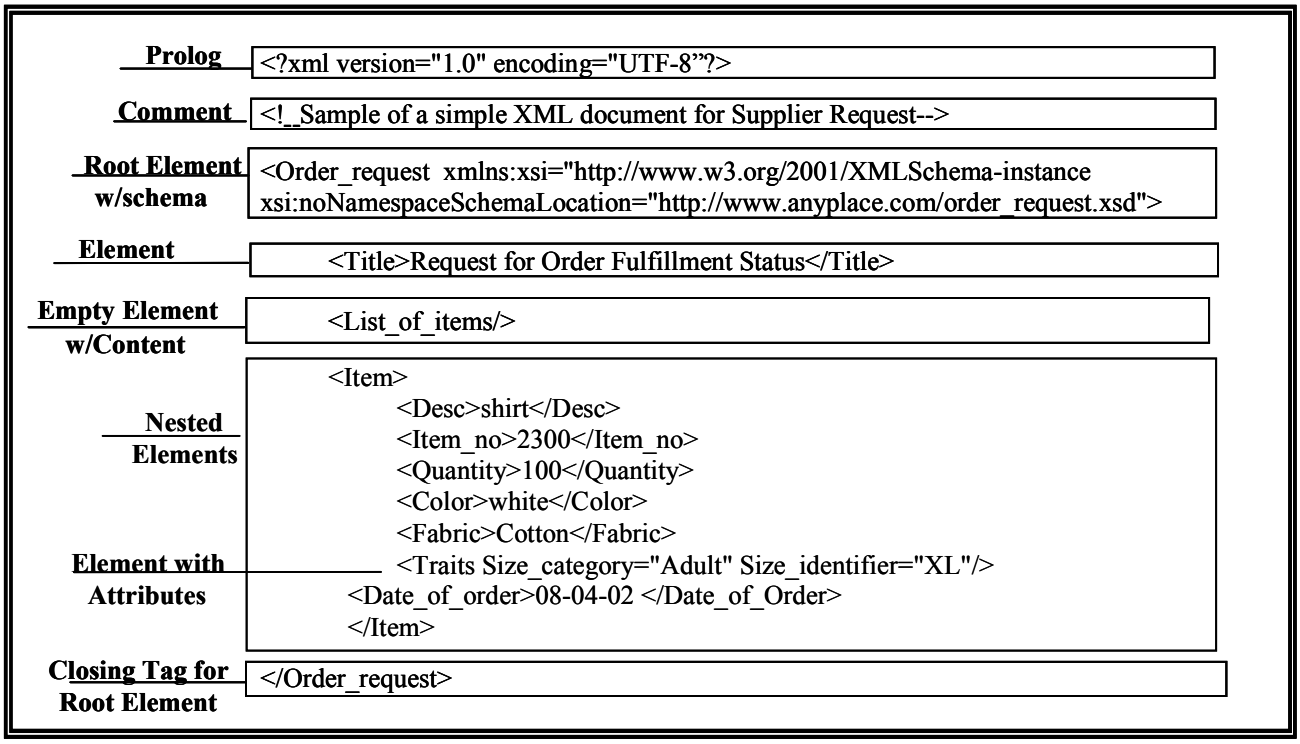

Figure1. Sample of a Simple XML Document

Empty Element

Elements with no content are known as empty elements. Empty elements are used when only the presence of the element is needed or the element contains only attributes.

Nested Elements

Nested elements may contain other elements. In Figure 1, the Item element contains several other elements such as Desc and Item_no. In fact, the basic structure of XML documents is based on nesting all elements within the root element. The improper nesting of elements is one of the most common mistakes made when developing XML documents. If an element starts within another element, it must also end within that element. Table 1 shows the difference between incorrect and correct placement of end tags.

Table 1. Example of Incorrectly and Correctly Nested Elements

Correctly Nested Elements	Incorrectly Nested Elements
<Item> <Desc>shirt </Desc> </Item>	<Item> <Desc>shirt </Item> </Desc>
	**The end tag for the Prefix element must come before the end tag for the Course element*

Elements with Attributes

Elements with Attributes - Attributes are included in the beginning tag of an element and follow the format attribute = value. The value must always be enclosed in single or double quotes. An element may contain several attributes such as in Figure 1 where the element Traits contains two attributes, Size_category and Size_identifier. The decision to use an attribute or element is a matter of preference. A general rule, however, is to use an element for content that needs to be extracted individually and attributes for content not independently relevant. For example, the attributes Size_cateory and Size_identifier are not independently relevant but are directly related to the complete identification of the size of the item.

Closing Tag for Root Element

Generally the last line of an XML document is the closing tag for the root element.

Once the XML document is created, it must be checked for well-formedness. The rules for a well-formed document include:

- The XML Declaration must appear at the beginning of the document.
- The Root or Document Element must contain all other elements. The only statements allowed to appear before the Root Element are the XML Declaration, comments, or processing instructions.
- All elements must have a start tag and end tag.
- Always begin and end tags and entities with the symbols < and > respectively.
- Empty elements must either end with the /> or have both the start and end tags.
- Elements must be properly nested.
- Attribute values must be enclosed in quotes with the double quotation mark (“) being the most commonly used.
- XML is case sensitive; NAME and name are not the same.
- Markup characters (< ,&,>, ",') cannot be used within XML content. Instead their corresponding entity reference must be employed. The predefined entities for the mark up characters are &lt, &amp, &gt, &quot and &apos.

Constructing the Schema

The schema defines and constrains the data that may be contained in an XML document. On advantage to a schema is that it provides several built-in datatypes as well additional derived datatypes. Some of the more commonly used datatypes include string, Boolean and those related to numbers (double, decimal, float, positiveInteger, etc.) and time. Figure 2 depicts the schema for the XML document listed in 1.

Description of the XML Schema Components

Root Element
The XML format requires that a root element be established. In the XML Schema, the root element is 'schema' with an attribute that identifies a namespace. The prefix xsd is mapped to the namespace and used throughout the schema document.

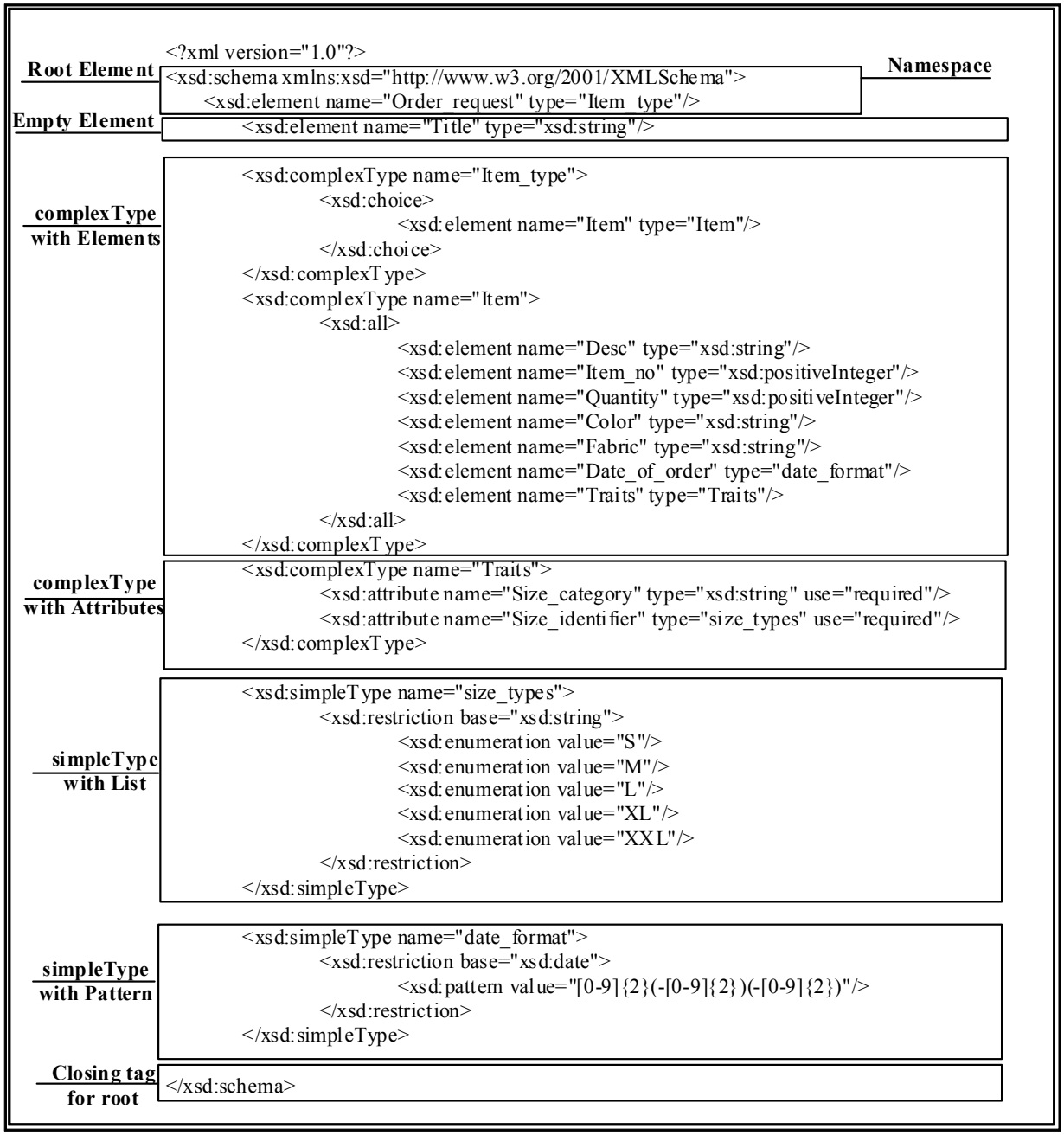

Figure 2. Sample of Simple Schema

Namespaces

One advantage to XML technologies is the ability to share XML applications. One problem, however, is that the same tag name may be used in both applications but applied differently. Namespaces provide a way to associate elements and attributes to a specific XML application by mapping them to a particular URL. The URL does not have to point to a DTD or schema document. The URL is simply applied as a prefix making element and attribute names unique. Multiple namespaces may be used in an XML document. The namespace declaration appears in the prolog of an XML file. The W3C schema specification, http://www.w3.org/2000/10/XML Schema, is used in almost every XML Schema.

XML Document

The namespace declaration appears in the prolog of an XML file. The W3C schema specification, http://www.w3.org/2000/10/MLSchema , is used in almost every XML Schema.

The second root element is the root element of the schema document and usually matches the root element in the XML document. The type is a new type declared later in the schema. The new type contains the element of 'Item.' The type is declared to be choice indicating the elements may appear in any order.

ComplexType

A schema may contain simpleType and complexType elements. A simpleType element is an element that only contains text and does not have any attributes or child elements. A complexType element may contain elements and/or attributes. The first complexType describes the elements contained within 'Item' and the second describes the attributes of 'Traits.'

SimpleType with Enumeration
The schema enumeration limits the element value to a choice from a list of specified values. Within this schema, enumeration is assigned to a simpleType named 'size_types' which is affiliated with the attribute 'size_identifier.'

SimpleType with Pattern
Pattern allows a format to be assigned to a value. Several options are available for pattern matching. In this schema, the simpleType is constrained to the format of two numbers, a dash, two numbers, a dash and two more numbers. To match the date datatype, an acceptable value would be 08-09-02 indicating month, day and year respectively. The simpleType, date_format is associated with the Date attribute of the Date_of_order element.

XML Example Scenario

The potential of XML is probably best explored through an illustration using a simple scenario. This scenario describes the use of XML to automate a common practice within e-commerce; that of negotiation between retailer and supplier. A retailer has a small online storefront that accepts orders from customers. As a small business, the retailer keeps a very limited supply of inventory on hand. When a large order is received, the retailer must contact suppliers to secure additional inventory. This order might be faxed, emailed or even given over the phone. This manual process is time consuming and often results in delays. An automated process would make the procedure more efficient. XML provides a solution. Figure 3 shows how XML might be used.

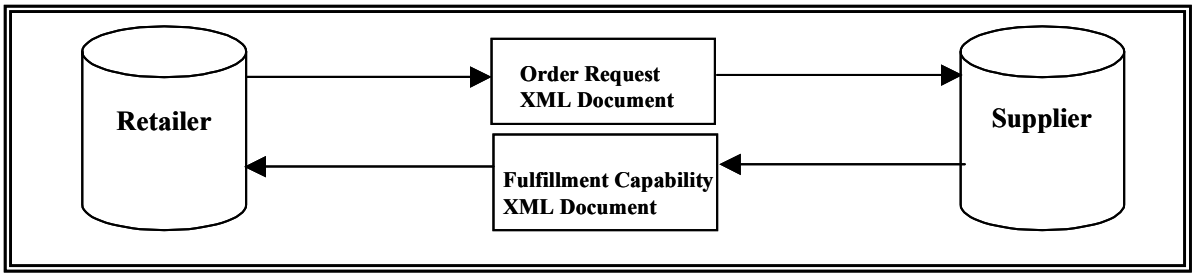

Figure 3. Applying an XML document as an Order Request

The retailer receives an order from a customer and enters the information into their database. Then the retailer extracts information related to inventory needed to complete the order (ie size, quantity, color, ect.) and writes this information to an XML document employing the appropriate tags for each item needed. The XML document file is then sent over the Internet to the supplier. The supplier reads the XML document, extracts needed data and compares it against their inventory database and creates a new XML document indicating how they can meet the retailer's request. This fulfillment capability XML document is then sent back over the Internet to the retailer who can then begin the process of completing the transaction for purchase of the goods.

However, for this process to be successful, there is a caveat. The retailer and the supplier have to agree upon the tags used in the XML document. Otherwise the process cannot be handled automatically. For example,, a name can be presented in several ways; First-name, Last_name and Middle_initial as three separate elements or as three attributes of one element. The XML document created by the retailer would not be usable by the supplier if the retailer chose to use attributes when the supplier was expecting elements. The XML solution to this problem is to develop a schema that is shared by both the retailer and supplier. (In fact, this schema can be shared with other suppliers as well). The schema provides the data definition for what can be contained in the XML document. Figure 4 shows the process when a schema is used.

The retailer has access to the common schema and now the data that is extracted from their internal database is written to an XML document that follows the definitions in the schema. Further, the XML document is validated against the schema to ensure compliance. The validated document is sent across the Internet to the supplier. The supplier knows exactly what data was sent because they also have access to the schema. Further, the fulfillment capability XML document created by the supplier in response to the retailer's request will also be understood by the retailer.

While this scenario is simple, it demonstrates the potential of using XML as a way to standardize the data exchange process. While there are other proprietary methods such as Electronic Data Interchange (EDI) currently in place that achieve similar results, these systems are generally expensive and only used by large organizations. XML technologies creates opportunities to share data regardless of organization characteristics.

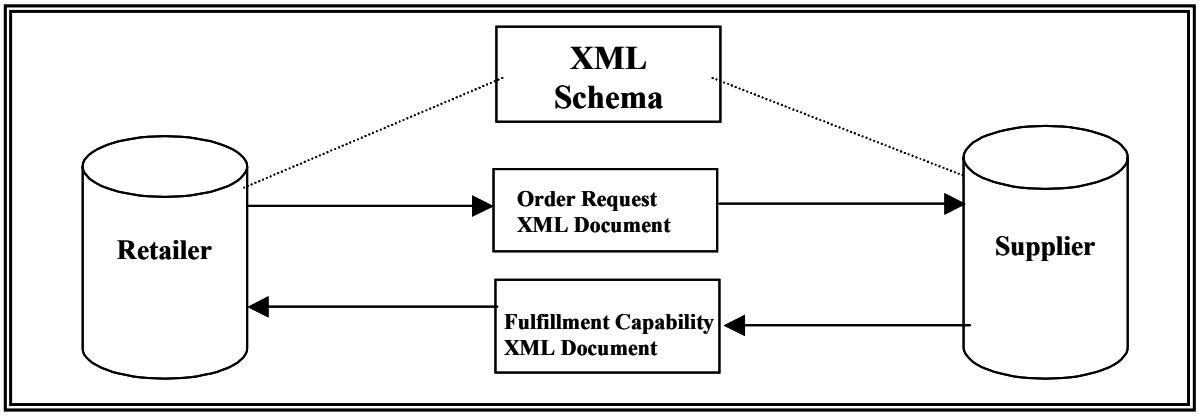

Figure 4. Applying the XML Schema to the Order Request

Conclusions

XML and its associated standards will impact the way systems are developed and the way organizations share information. The primary challenges will be in the development of common vocabularies and standardization of business processes. Consequently, implementation will not be revolutionary; it requires an evolutionary approach that includes introspection of internal data models and reconciliation with data models from organizations to which information will be exchanged. XML should not be viewed as just another Web technology; but as a foundational technology that will transform the IT infrastructure.

References

Harold, E. R. *XML Bible: Second Edition.* New York: Hungry Minds, 2001

Herman, J. The XML Internet takes off. *Business Communications Review*, 31(4), April 2001, p.27.

Patrizio, A. XML passes from development to implementation. *Information Week*, 830, April 26, 2001, 116-120.

Shirky, Clay. XML: No Magic Problem Solver. *Business 2.0*, 1, September 26, 2000, 75.

Waldt, D. and Drummond, R. The Global Standard for Electronic Business. http://www.ebxml.org/presentations/global_standard.htm, retrieved April 23, 2002

Whalin, D. (2002). Top Five Uses for XML. *XML Magazine*. http://www.fawcette.com/xmlmag/2002_01/online/online_eprods/xml_dwahlin01_18/default.asp, retrieved April 23, 2002.

World Wide Web Consortium, Extensible Markup Language (XML) 1.0, http://www.w3.org/XML/ , retrieved April 23, 2002.

World Wide Web Consortium (2001, December, 13). XML: Working Draft 13 December 2001. http://www.w3.org/TR/xml11/ Retrieved April 23, 2002