Communications of the Association for Information Systems

Volume 9 Article 7

September 2002

Using the Extensible Markup Language (XML) As a Medium for Data Exchange

Meg Murray

Kennesaw State University, mcmurray@kennesaw.edu

Communications of the Association for Information Systems

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

Meg Murray
Department of Computer Science and Information Systems
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. XML is being adopted rapidly, and 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. The power behind XML is its simplicity; however there is still much confusion about this technology. XML will evolve as its structure, its strengths, its weaknesses, and how it can be used more effectively are better understood.

This paper includes an overview of XML and XML specifications and corresponding components, technical implementation requirements, the development of schemas for defining industry standard data definitions, a scenario employing XML technologies, and a discussion of the potential impact of XML on information systems.

KEYWORDS: XML, extensible markup language, data exchange, emerging technologies

I. 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 implemented easily 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 for data exchange.

XML is part of a 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].

Much hype surrounds XML with predictions that XML will replace currently used Web technologies such as HTML and even Java. The truth is that while XML will transform various activities that take place over the Internet, XML is more of an enabling technology that complements existing tools. Currently, the primary role XML serves is that of a standard method for sending and exchanging information over the Internet. XML will evolve as its structure, its strengths, its weaknesses, and how it can be used more effectively are better understood.

The power behind XML is its simplicity. XML separates data from its presentation and processing. There has been confusion about what this separation means. Usdin and Graham [1998] state that: "For all practical purposes, XML is also a philosophy of data identification and a host of languages and specifications for functions related to data" [p. 125]. The foundation of XML is a document that contains data embedded in tags that identify the data. Unlike other familiar markup languages, such as HTML, the tags within XML are created by the user. This ability means that XML may be customized to meet the needs of particular applications [Usdin and Graham, 1998]. A custom XML tag set allows users to identify information and manipulate the data in different ways depending on the need. Associated XML specifications continue to be defined to facilitate this manipulation of data. Three common specifications include:

- the Document Type Defintion (DTD),
- the schema (discussed later) and
- the eXtensible Stylesheet Language (XSL).

XSL is used to create stylesheets to render or present data contained in an XML document. XSL contains a set of rules that are used to build templates that specify how XML document data are to be formatted for display. XSL is still under development. Consequently, no browser currently in use fully supports XSL. The current approach is to use an XSLT processor to transform XSL stylesheets into HTML output. Other activities surrounding XML specifications are numerous. Some of the notable ones include

- Xlink and Xpoints that support multi-directional links and links to individual parts of a document,
- Xforms an enhancement to HTML forms, a Web services interface specification,
- SOAP an XML-based protocol for exchanging messages and
- Xquery which is an XML-based database query language.

Interest in XML is high, but the newness of the technology and the evolving nature of XML standards means that a learning curve surrounds XML technologies. Even given these shortcomings, XML is fast becoming a transformational technology. XML is being adopted at a rapid rate and is 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 evolving and 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 that 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 implemented widely.

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].

The simplicity of XML opened this technology to limitless possibilities for its use. Specialized tag sets defined in community schemas are being developed that span a wide range of applications from facilitating specialized symbols sets as mathematical and chemical symbol manipulation and presentation to VoiceXML for synthesizing speech, to using XML as a means for standardizing the exchange of information to complete business processes in a wide array of industries, including e-commerce, law and healthcare. Some of the earliest uses of XML were for document manipulation. 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
- managing computer application configuration data [Wahlin, 2002].

The focus of this paper is on the use of XML for data exchange.

II. 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’, that is, a document that includes information describing what the document contains. As XML is 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 stylesheet 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 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 character data often denoted as the datatype of 'string.' Using a DTD 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 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 exist. Parsers are included with commercial XML development environments, included with books on XML, and can be found on the Internet.) The parser identifies any error found but does 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.

III. 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 that comes with Microsoft's Windows™ operating systems. (Available commercial XML editors provide such support as automated formatting and automatic entry of tag symbols.) 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 paper 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.

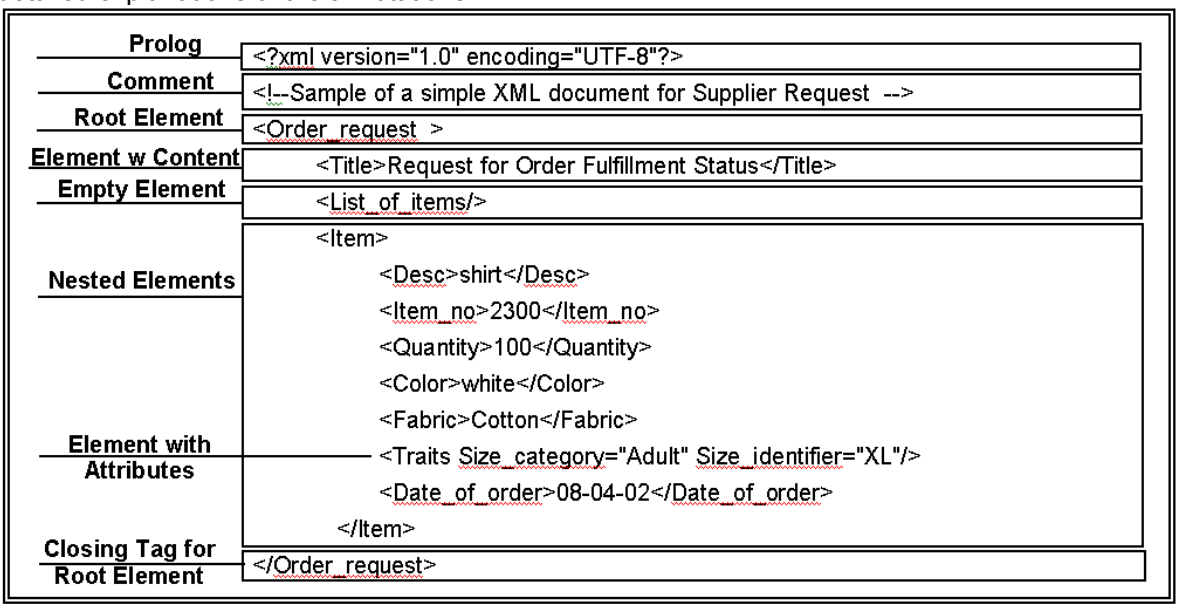

Figure 1. Sample of a Simple XML Document

DESCRIPTION OF THE XML DOCUMENT COMPONENTS

Prolog

The prolog contains the XML Declaration, which is always the first line of any XML file. Nothing, not even white space, should be put 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 used (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

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 root element may also include attributes that point to associated XML specifications such as associated schemas or stylesheets. Attributes will be demonstrated at the end of Section IV.

ELEMENT WITH CONTENT

The most common format for elements is: <start-tag> content </end-tag>. This format is used for the majority of elements in Figure 1. The element names contained within the tags are created by the developer. They must adhere to specific naming rule. Naming rules and common guidelines include:

- All element names used within an XML document must be unique
- Element names may not contain spaces
- Element names must not start with a number or a punctuation/special symbol mark
- Element names may not begin with the letters 'xml'
- Avoid using '-' and '.' and ':' in element names
- Element names are not restricted by length but long names can be confusing
- Choose names that are descriptive of the data
- If the data matches data contained in a database, it can be useful to name elements with the same name as the corresponding database field name.

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. Note than an empty element may be written combining the beginning and end tags, <Empty_Element/>, as is done in Figure 1.

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 'Desc' element must come before the end tag for the 'Item' element

ELEMENTS WITH ATTRIBUTES

XML supports the use of attributes. Attributes are contained within the element tag and provide additional information about the element. 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 often a matter of preference. The choice to be made is between using nested or child elements or using an element that contains attributes. For example, the Traits element could also be written as:

```
<Traits>
        <Size_category>Adult</Size_category>
        <Size_identifier>XL</Size_identifier>
</Traits>
```

A general rule, however, is to use an element for content that needs to be extracted individually and attributes for content not relevant independently. For example, the attributes Size_cateory and Size_identifier are not independently relevant but are directly related to the complete identification of the traits of the item. A disadvantage to using attributes is that an attribute may not contain other attributes whereas an element may contain child elements. The decision should be made based on need; elements provide more flexibility while attributes help to clarify element characteristics.

CLOSING TAG FOR ROOT ELEMENT

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

CHECKING FOR WELL-FORMEDNESS

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.

VIEWING THE XML DOCUMENT IN A BROWSER

Web browsers that support XML do not understand the tags in any given XML document (unless the tags happen to duplicate an HTML tag). The style of presentation must be defined for the document, often by using a stylesheet language such as XSL described in Section I. If no style is applied to an XML program, the browser will only show the text of the document. The browser will provide expanding and collapsing capabilities for nested items. Figure 2 shows the browser display for the XML document presented in Figure 1 with the nested element ‘Item’ collapsed.

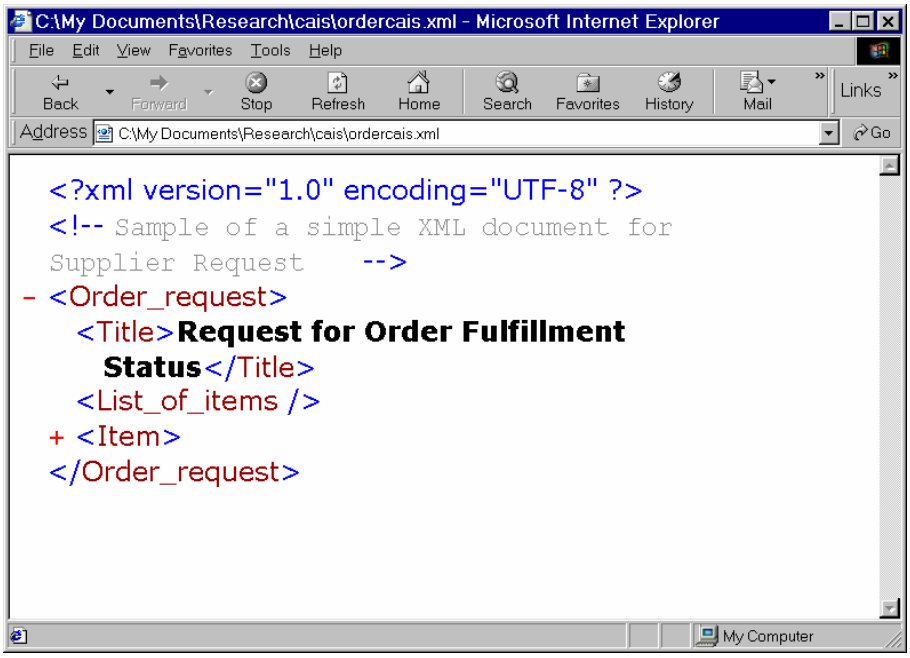

Figure 2. Browser Display of an XML document

CONSTRUCTING THE SCHEMA

The schema defines and constrains the data that may be contained in an XML document. It provides a way to specify

- what tags may be used,
- what data those tags identify,
- what characteristics the data may have and
- what range of values the data may contain.

Further, the schema provides a common data definition and can be shared among many XML documents regardless of their location. The schema is created using the XML Schema Definition Language, a recommendation of the W3C. One advantage to a schema is that it provides several built-in datatypes as well as additional derived datatypes. Some of the more commonly used datatypes include

- string,
- Boolean
- those related to numbers (e.g., double, decimal, float, positive Integer), and
- time.

Figure 3 depicts the schema for the XML document listed in Figure 1.

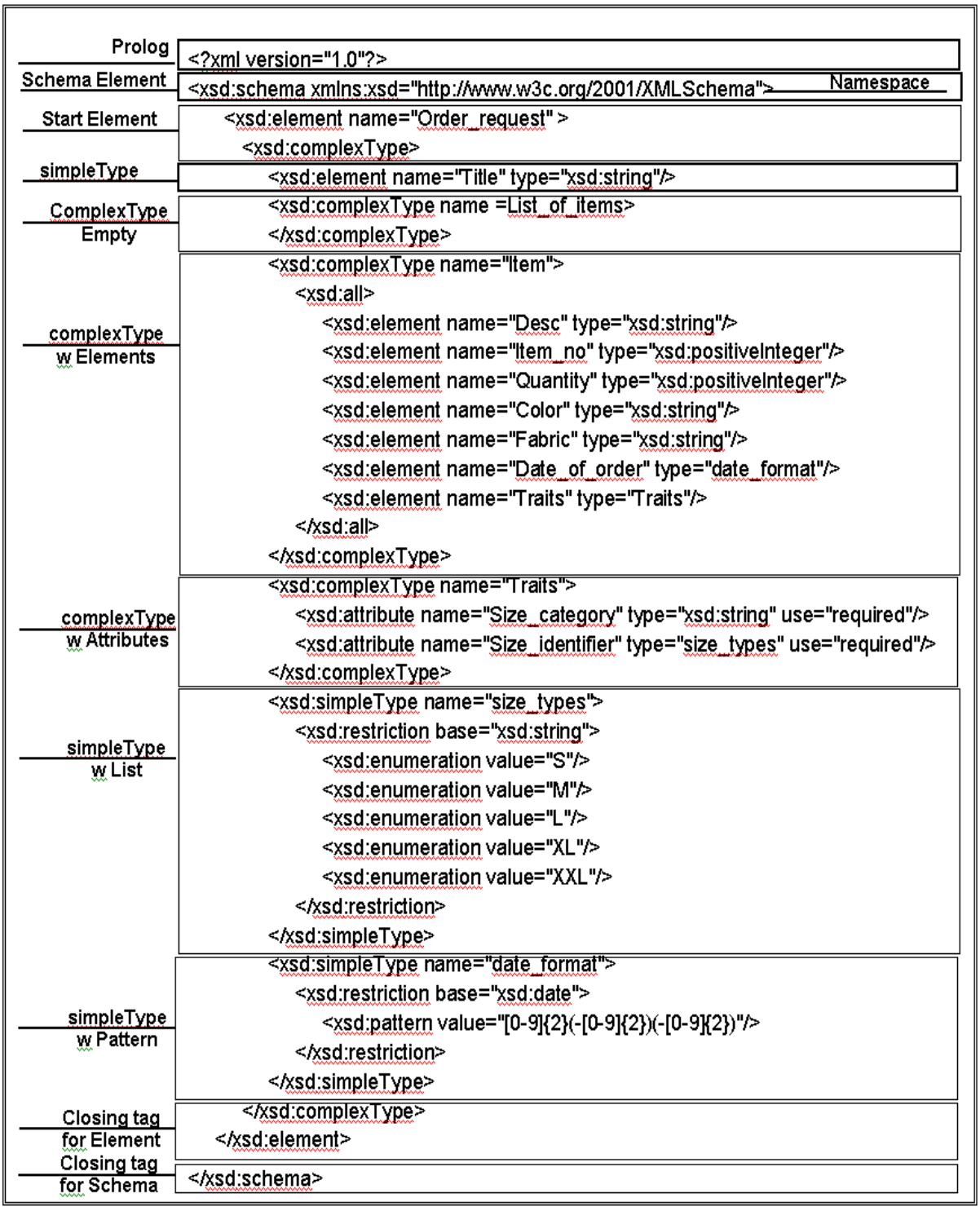

Figure 3. Sample of a Simple Schema

DESCRIPTION OF THE XML SCHEMA COMPONENTS

Prolog

Prolog is the XML Declaration just as is used in the XML document.

Schema 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.

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 may or may not point to a schema document or any other file for that matter. The URL is simply applied as a prefix that makes element and attribute names unique. However, to use definitions defined in other schemas, the namespace must point to an existing document. Almost every XML schema will point to the W3C schema specification, http://www.w3c.org/2001/XMLSchema , referred to as the 'Schema of all schemas.' Among other things, this Schema contains the definitions for the predefined datatypes. Multiple namespaces may also be used in an XML document. They are assigned a prefix (xsd in Figure 3) used to identify elements that reference that schema's instructions.

Matching Start Element

The next element matches the start (root) element tag from the associated XML document file. In Figure 3, it is the 'Order_request' element. Note the syntax for elements. They are named using the form: element name = "associated XML document tag name placed within quote marks". 'Order-request' is defined as a complexType element. All additional elements from the associated XML document will be nested within this element.

COMPLEXTYPES AND SIMPLETYPES

A schema may contain simpleType and complexType elements. A simpleType element is an element that only contains text and does not have attributes or child elements. A complexType element may contain elements and/or attributes.

SimpleType

SimpleType elements are defined specifying the element name and its datatype. The datatype may be one that is defined in the current schema or in an external schema such as those predefined by the W3C. The schema for externally defined datatypes must be specified as a namespace attribute in the schema element. This was done for the W3C schema as demonstrated under the 'Schema Element' label in Figure 3. The datatype will be preceded by the appropriate prefix (xsd in Figure 3) if referenced in an external schema. The first example of a simpleType shown in Figure 3, is an element with the name 'Title' and the predefined of datatype 'string.'

ComplexType with Elements

Three additional complexType elements are defined in Figure 3. The first one, 'List_of_Items' demonstrates how an empty element would be defined. 'Items' demonstrates a complexType containing elements and 'Traits' demonstrates a complexType with attributes. The child elements contained within the element 'Item' are enclosed within the *all* compositor. Compositors within a schema allow the order of sub-elements to be specified; this order relates to how data must be included in the associated XML document. There are 3 compositors:

- sequence which specifies that all the named elements must appear in the specified order
- choice which specifies that one and only one of the elements listed must appear

- all which specifies that all elements must appear but they may appear in any order.

ComplexType with Attributes and Defined Datatype

Elements with attributes may also be defined. The general rule is that elements with attributes are placed after elements without attributes within the complexType listing of elements. In Figure 3, the attributes are defined with a name, datatype, and use. Use can be identified as optional, prohibited, or required. The first attribute, 'Size_category' is a required attribute of datatype string. The second attribute, 'Size_identifier' is also required but its datatype is defined within this schema. The datatype 'size_types' appears later in the schema after all elements and attributes are defined.

SimpleType with List (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 a defined datatype affiliated with the attribute 'size_identifier.' 'Size_types' is defined as a datatype restricted to characters (strings) which are limited to the values (S, M, L,XL, or XXL).

SimpleType with Pattern

Patterns allow a format to be assigned to a value. Several options, as defined in the W3C XML Schema specifications [W3C, 2001], are available for pattern matching. In this schema, the simpleType is restricted to a date value and 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.

Closing Tags for Element and Schema

To maintain wellformedness, the start element (and its type because it appears as its own tag) and the schema element must be closed to ensure the integrity of the nesting of all elements.

REFERENCING THE SCHEMA IN THE XML DOCUMENT

Schemas are associated with an XML document. Data contained within the XML document are validated against the criteria specified in the schema. Several validation programs are available. Many are listed on the W3C XML resources web site: (http://www.w3c.org/XML/Schema#resources).

Schemas can be included directly in an XML document but it is more common for the schema to reside in its own file. This arrangement provides more flexibility. The schema may be located anywhere on the Internet as long as it is accessible. In this case, the schema is referenced from within the XML document as an attribute of the root element. (An XML document may reference multiple schemas.) Figure 4 shows the XML document depicted in Figure 1 with the inclusion of references to two schemas:

- The first schema, ' xmlns:xsi = http://www.w3.org/2000/XMLSchema-instance ', is a standard schema defined by the W3C. Elements that reference this schema will not be preceded by the prefix 'xsi'. In Figure 4, it is only applied once.
- The schema

  'xsi:noNamespaceSchemaLocation=http://www.anyplace.com/order_request.xsd '

  points to the location of a schema that describes the data contained in this XML document. The 'noNamespaceSchemaLocation=' means that elements from this schema do not need to include a prefix. The use of prefixes is what allows multiple schemas to be included in one XML document. The general rule is that a prefix not be designated for one schema, but a unique prefix be assigned to all other schemas referenced within the XML document.

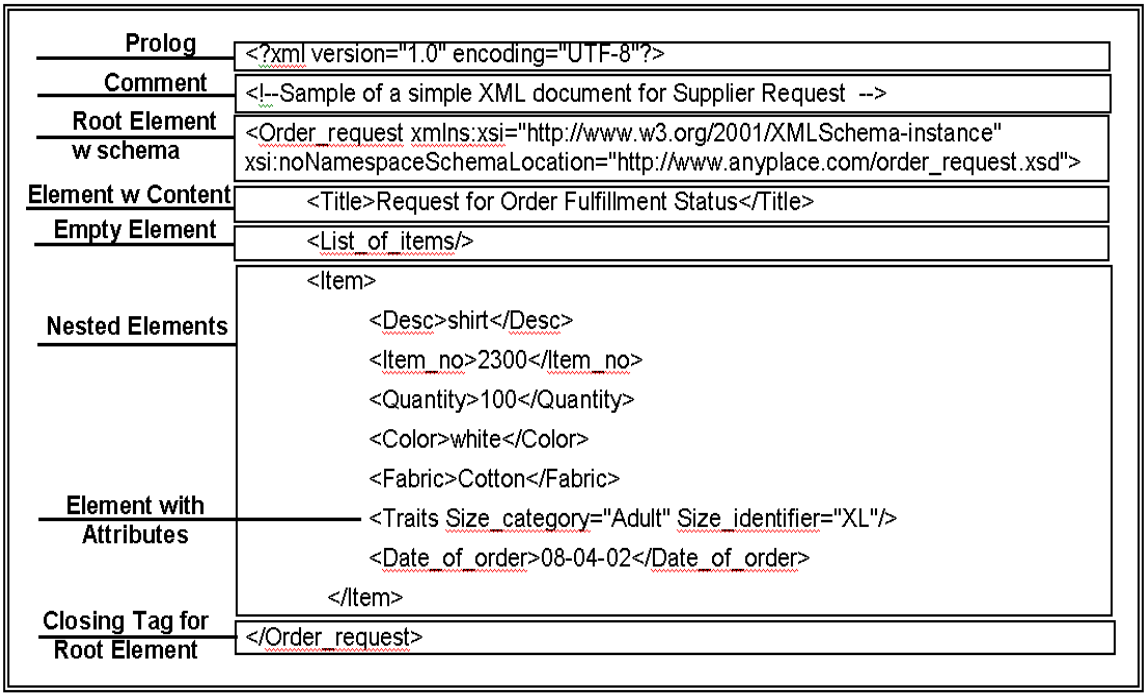

Figure 4. XML Document with Reference to a Schema

V. 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 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 5 shows how XML might be used.

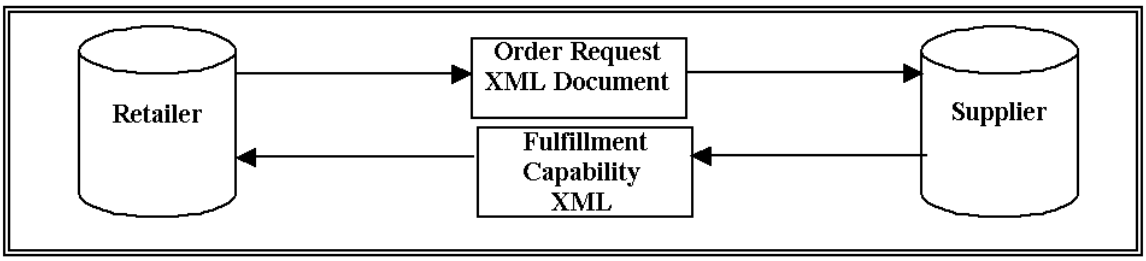

Figure 5. Applying an XML document as an Order Request

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 (e.g., size, quantity, color.) 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 own 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, there is a caveat. For this process to be successful, the retailer and the supplier must 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 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 6 shows the process when a schema is used.

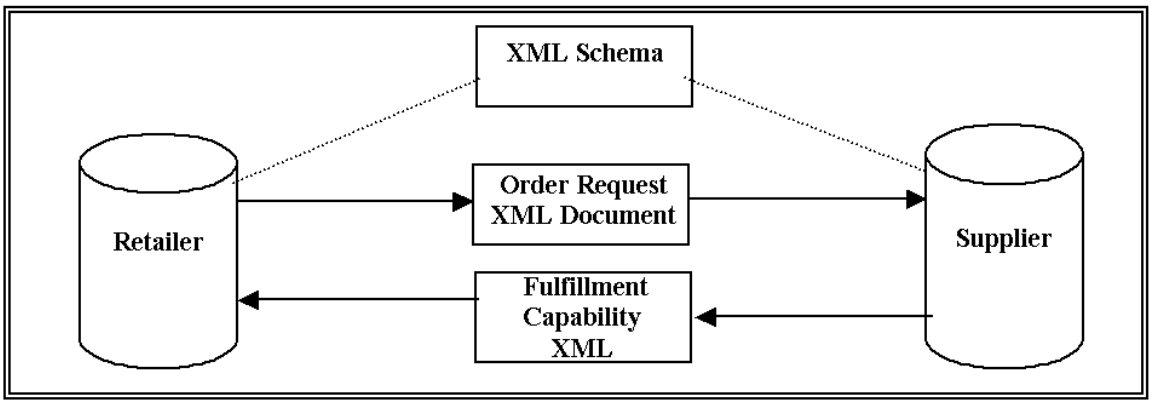

Figure 6. Applying the XML schema to the Order Request

Using the common schema, the retailer extracts the data from their internal database and writes it to an XML document. 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 access the same schema. Further, the fulfillment capability XML document created by the supplier in response to the retailer's request is also 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 other proprietary methods such as Electronic Data Interchange (EDI) are currently in place that achieve similar results, these systems are generally expensive and are only used by large organizations. XML technologies create opportunities to share data regardless of organization characteristics.

VI. CONCLUSIONS

XML and its associated standards will impact the way systems are developed and the way organizations share information. However, while creating an XML document is a fairly easy concept to understand, extending the usability of XML by applying associated specifications, such as those provided by the Schema Specification Language, is more complex. Other hurdles to overcome include building consensus for common data models, vocabularies and standardization of business processes. Consequently, implementation of XML technologies will not be revolutionary; it will require an evolutionary approach that includes introspection of internal data models and reconciliation with data models from organizations with which information will be

exchanged. The adoption of XML will be progressive, but its importance must not be understated. XML is not just another Web technology; XML is a foundational technology that will transform the IT infrastructure.

Editor's Note: This article was received on August 26, 2002 and was published on September 12 , 2002.

REFERENCES

EDITOR'S NOTE: The following reference list contains the address of World Wide Web pages. Readers who have the ability to access the Web directly from their computer or are reading the paper on the Web, can gain direct access to these references. Readers are warned, however, that

1. these links existed as of the date of publication but are not guaranteed to be working thereafter.
2. the contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions referenced.
3. the authors of the Web pages, not CAIS, are responsible for the accuracy of their content.
4. the author of this article, not CAIS, is responsible for the accuracy of the URL and version information.

Herman, J. (2001). "The XML Internet Takes Off," *Business Communications Review*, 31(4), pp. 27.

Patrizio, A. (2001, April 26). "XML Passes from Development to Implementation," *Information Week*, 830, pp. 116-120.

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

Usdin, T. and Graham, T. (1998) "XML: Not a Silver Bullet, but a Great Pipe Wrench", *Standardview*, (6)3, pp. 125-132.

Waldt, D. and Drummond, R. "The Global Standard for Electronic Business," http://www.ebxml.org/presentations/global_standard.htm (current August 23, 2002).

Weiss, A. (1999). "XML Gets Down to Business," *NetWorker*, 3(3), pp. 36-43.7

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 (Current August 23, 2002).

World Wide Web Consortium,(2000, October 6). "Extensible Markup Language (XML) 1.0," http://www.w3.org/XML/ (Current August 23, 2002).

World Wide Web Consortium, (2000). "W3C Architecture Domain: XML Schema," http://www.w3c.org/XML/Schema#resources (Current August 23, 2002).

World Wide Web Consortium (2001, December, 13). "XML: Working Draft 13 December 2001," http://www.w3c.org/TR/xml11/ (Current August 23, 2002).

BIBLIOGRAPHY

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

Castro, E. (2001). *XML for the World Wide Web*. Berkeley, CA: Peachpit Press.

van der List, E. (2000). "Using W3C XML Schema," http://www.xml.com/pub/a/2000/11/29/schemas/part1.html (Current August 23, 2002).

W3C (2001). "XML Schema Part 0: Primer," http://www.w3c.org/TR/xmlschema-0/ (Current August 23, 2002).

W3C (2001). "XML Schema Part 1: Structures," http://www.w3c.org/TR/xmlschema-1/ (Current August 23, 2002).

W3C (2001). "XML Schema Part 1: Datatype," http://www.w3c.org/TR/xmlschema-2/ (Current August 23, 2002).

ABOUT THE AUTHOR

Meg Murray is a faculty member in the Department of Computer Science and Information Systems at Kennesaw State University, part of the higher education system of the state of Georgia. She received a Ph.D. in Information Systems, a MBA in Management and a MS in Computer Science. She has been in the field of computing for more than twenty years in both higher education and industry. She specializes in emerging technologies and the development and implementation of those technologies to meet business and organizational needs. Current publications deal with the implementation of decision support and expert systems to support healthcare and with the development of web services including exploring the supporting technologies of XML, SOAP, .Net and J2EE. Her research interests continue to be in the area of emerging technologies and their impact on changing business models.

Communications of the Association for Information Systems

ISSN: 1529-3181