IMPLEMENTING AN XML COURSE IN THE COLLEGE OF BUSINESS

Thom Luce
Ohio University
MIS Department
luce@ohio.edu

ABSTRACT

Over the past decade much of computing moved from mainframe centric systems to client-server systems and from client-server to Internet/Web based systems. The move to the Web and to HTML based applications made communication between widely distributed systems easy – at least the display of information was easy, not necessarily the effective and accurate distribution of information because HTML primarily describes the structure of a document, not the structure of the data in the document.

XML, the Extensible Markup Language, was developed by the World Wide Web Consortium (W3C) to solve this problem. XML is a meta-language for the creation of other languages, specifically languages that describe the structure of data in an unambiguous way along with languages that specify how to display and transform data.

XML is especially interesting and applicable to business because it allows for the development of customized applications and standardization of data sharing in almost any industry. The potential of these applications are significant enough to cause companies like IBM, Sun and Microsoft to bet major portions of their future on XML based technologies. XML is a fundamental technology underlying Microsoft's .NET initiative and a high degree of support for XML documents is already built into Microsoft Internet Explorer version 5.0 and above.

This paper looks at the incorporation of XML into the MIS curriculum at Ohio University where it was first introduced as one part of a quarter long advanced web development course and then expanded to a full quarter advanced topics course. The paper describes topics covered in the course with samples of assignments given. The paper concludes with a discussion of the importance and application in the business school curriculum.

KEYWORDS

XML, Undergraduate courses, Internet, Web development

INTRODUCTION

Much of computing moved from mainframe centric systems to client-server systems during the 1980s and then from client-server to Internet/Web based systems in the 1990s. The move to the Web made communication between widely distributed systems easier than it had ever been before because of the use of a “common” language – HTML.

As we quickly discovered, HTML made it relatively easy to display information on a wide range of devices, but it isn't very effective or accurate when it comes to the distribution of information because HTML primarily describes the structure of a document, not the structure of the data in the document.

To solve these problems The World Wide Web Consortium (W3C) and its member organizations developed XML, the Extensible Markup Language as a standard for describing data (1). Steve Ballmer, Chief Executive Officer of Microsoft Corporation described XML as the "lingua franca of the Internet" and believes that the "XML Revolution" will be "as bigger or even bigger than any revolution that preceded it (2)."

XML is a meta-language for the creation of other languages, specifically languages that describe the structure of data in an unambiguous way along with languages that specify how to display and transform data. Version 1.0 of the XML standard, or Recommendation as the W3C calls it, was released in February 1998 with a second edition approved in October 2000.

To create the consistency necessary for the accurate exchange of data over the Web, the structure of XML documents may be defined with a Document Type Definition (DTD) or an XML Schema, which achieved Recommendation status in May 2001. As Tim Berners-Lee, the original developer of the World Wide Web and the current Director of the W3C says, XML Schema is "an XML language for defining XML languages (3)." Even more recently, specifications for XML Base and XML Linking Language (XLink) were released as Recommendations. XML Base is used to establish a URL base for XML document linking and XLink permits linking to internal and external resources (analogous to \ <base> and <a href> \ in HTML) (4). Additional components of XML include XSL (Extensible Stylesheet Language, a Candidate Recommendation in November 2000), XSLT (XSL Transformations, Recommendation released in November 1999) and XPath (the XML Path Language used by XSLT to locate different parts of an XML document, Recommendation release in November 1999).

XML is a very flexible markup language; it has no predefined tags and only a few simple rules (there must be one root element, only comments, and processing instructions, are allowed outside the root element, closing tags are always required, elements must be properly nested, case matters, all values must be enclosed in quotes, and a few others.) This simplicity, while a strong point of XML, is also one of its major weaknesses – anyone can create markup tags and create a valid XML document. The usefulness of XML as a way to describe data requires agreement among interested parties on the tags used (5).

To that end, numerous groups are working on the development of common mark up tags and schema for different applications. A few of the organizations involved in these efforts include the Institute of Certified Public Accountants (developing an Extensible Business Reporting Language, or XBRL) (6), the XML ISO 15022 Committee Advisory Group (trying to create a single standard from the various financial standards already developed) (7), the Open Travel Alliance, or OTA (working on an XML standard for sharing electronic ticket information) (8), and the National Retail Federation, or NRF (working to use XML for updating store inventories and for performing price checks) (9). There is even an attempt by the Organization for the

Advancement of Structured Information Systems, OASIS, to avoid the November 2000 election problems with the development of an Election Markup Language (EML) (10).

The rapid explosive growth of XML has caused problems for educators. Is XML something that will be here for the long run or just another technology *du jour*? Should we teach the technology in an AACSB accredited program? What should we teach? How do we get a course approved through the curriculum process quickly enough to be useful?

The MIS program at Ohio University decided that the time was right for XML at the end of the 1999-2000 school year. Fortunately, we had an open-ended seminar course (MIS491) on the books that could accommodate XML without the red tape of the University curriculum approval process. XML was first introduced as part of an advanced Web development course during the fall and winter quarters of 2000-2001 and expanded to a full quarter course for the spring, 2001 quarter.

MIS491 was offered as an elective course for MIS majors. Students taking the course had previously completed quarter long courses in both Visual Basic and Java and had exposure to VBScript and Active Server Pages. They typically did not have prior formal exposure to JavaScript. In addition to MIS majors, almost half the students enrolled in the class for the spring were graduate students from engineering because this was the first XML class offered at Ohio University.

COURSE STRUCTURE

The primary goal of the course was to give the students as much hands-on exposure to XML as possible in a ten-week quarter using currently available shareware and free tools such as XMLwriter, XML Spy, Microsoft Internet Explorer 5.x, Microsoft's msxml.exe parser, and java archive files (crimson.jar, jaxp.jar, xalan,jar and xerces.jar) from Sun, W3C and the Apache XML Project.

To do this, students were first given a brief introduction to XML, the rules used to create XML documents and were asked to create several sample documents. Figure 1 shows one of the initial assignments.

Figure 2 shows part of an XML document created for this assignment. The figure also shows Internet Explorer 5.0's default style for displaying XML files.

The next assignment required that the student take control of the display through the use of cascading style sheets. As with HTML tags, each XML tag can be formatted with the use of appropriate CSS settings. Figure 3 shows a formatted version of the XML file from Figure 2 and part of the CSS file used to create the formatting.

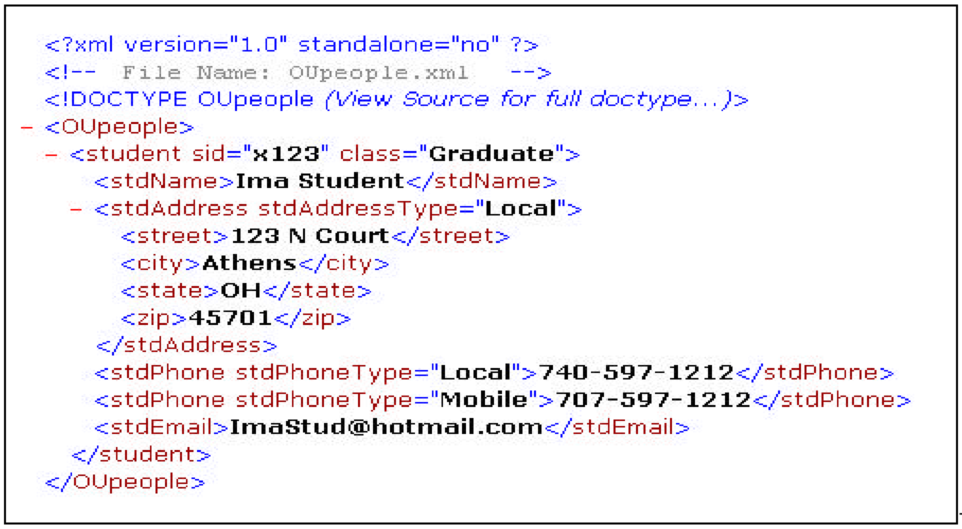
The next series of assignments introduced the student to the creation of schema definitions for their XML document. Students were required to write three different schema definitions for their OUpeople file: a Document Type Definition, or DTD, (because they are still widely used); a Microsoft XML Schema definition (Microsoft developed their initial parser on an early draft of the XML Schema document); and a definition based on the final XML Schema Recommendation. Figure 4 shows a sample DTD definition for <OUpeople>.

Create an XML document that holds information about people at Ohio University. The root element should be called "OUpeople" and it should contain separate elements for students, faculty and staff, in any order.

Student elements should have a required social security number attribute that is identified as an ID. Students should also have an attribute indicating their current class standing. This should be required and be one of the following: Freshman, Sophomore, Junior, Senior, Grad, Unclassified. The Student element should contain the following elements (in order):

- o Name
- o Address (with a required attribute indicating if it is a local or permanent address) - there may be one or more of these and you may define child elements.
- o Phone number (zero or more with an optional attribute indicating: local, permanent, mobile)
- o e-mail address (1 or more)

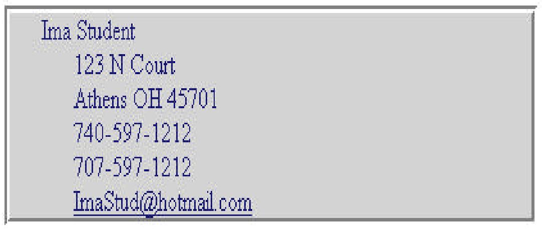
Figure 1. Sample assignment.



```
  <?xml version="1.0" standalone="no" ?>
  <!--  File Name: OUpeople.xml   -->
  <!DOCTYPE OUpeople (View Source for full doctype...)>
- <OUpeople>
  - <student sid="x123" class="Graduate">
      <stdName>Ima Student</stdName>
    - <stdAddress stdAddressType="Local">
        <street>123 N Court</street>
        <city>Athens</city>
        <state>OH</state>
        <zip>45701</zip>
      </stdAddress>
      <stdPhone stdPhoneType="Local">740-597-1212</stdPhone>
      <stdPhone stdPhoneType="Mobile">707-597-1212</stdPhone>
      <stdEmail>ImaStud@hotmail.com</stdEmail>
    </student>
  </OUpeople>
```

Figure 2. XML file displayed in IE 5.5.

After examining the structure of XML documents and how to develop XML schema, we turned to programmatic manipulation of the documents using the XML Document Object Model (DOM). The DOM is a W3C standard for representing and manipulating an XML document and DOM-based parsers such as msxml (Microsoft), JAXP (Sun), and Xerces (Apache) provide programmatic access to XML documents through a DOM Application Program Interface (API).



Ima Student
123 N Court
Athens OH 45701
740-597-1212
707-597-1212
ImaStud@hotmail.com

```
<style>
OUpeople {   BORDER: solid;
    DISPLAY: block }
student
{   BACKGROUND-COLOR: lightgrey;
    BORDER: groove;
    COLOR: midnightblue;
    DISPLAY: block;
    MARGIN-LEFT: 25px }
stdName
{   DISPLAY: block;
    MARGIN-LEFT: 25px }
stdAddress
{   DISPLAY: block;
    MARGIN-LEFT: 50px }
<!-- sections omitted -->
stdEmail
{   DISPLAY: block;
    MARGIN-LEFT: 50px;
    TEXT-DECORATION: underline }
</style>
```

Figure 3. XML file formatted with Cascading Style Sheet and part of the CSS used for the formatting.

```
<!-- File Name: OUpeople.dtd -->
<!ELEMENT OUpeople (student*, staff*, faculty*)*>
<!-- Student Element and its units -->
<!ELEMENT student (stdName, stdAddress+, stdPhone*, stdEmail+)>
<!ATTLIST student sid ID #REQUIRED>
<!ATTLIST student class
 (Freshman|Sophomore|Junior|Senior|Graduate|Unclassified)
 "Unclassified">
<!ELEMENT stdName (#PCDATA)>
<!ELEMENT stdAddress (street, city, state, zip)>
<!ATTLIST stdAddress stdAddressType (Local|Permanant) #REQUIRED>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT stdPhone (#PCDATA)>
<!ATTLIST stdPhone stdPhoneType (Local|Permanant|Mobile) #IMPLIED>
<!ELEMENT stdEmail (#PCDATA)>
```

Figure 4. DTD definition of <OUpeople> and <student>.

The DOM API provides classes and interfaces for various XML objects including the entire document, nodes, elements, attributes and text. It also provides methods for the creation, manipulation and deletion of different parts of an XML document including elements, attributes, text, and comments as well as methods for navigation (getDocumentElement, getParentNode, firstChild, lastChild, nextSibling, previousSibling, etc.).

All students in MIS491 learned to manipulate XML documents via the DOM API in JavaScript and Java and some students also used VBScript in Active Server Pages applications. Figure 5 shows a portion of the JavaScript code used to manipulation OUpeople.xml (the code displays the name of each student found in the file along with a list of any and all phone numbers associated with each student).

We also used the Java DOM APIs to manipulate XML documents and transfer them between computers using the XmlMessenger example found in the course textbook, *XML How To*

Program (11). The socket-based communication demonstrated by this example was used as the basis for several student projects at the end of the course.

```
forStudent.innerHTML = "Student List: <BR>";
forStaff.innerHTML = "Staff List: <BR>";
forFaculty.innerHTML = "Faculty List: <BR>";
var xmldoc, nodeList, OUpeopleNode;
xmldoc = new ActiveXObject("Microsoft.XMLDOM");
xmldoc.load("OUpeopleW3C-S.xml");
OUpeopleNode = xmldoc.documentElement;
var studentNode, studentList, staffNode, staffList
var facultyNode, facultyList;
studentList = OUpeopleNode.getElementsByTagName("student");
for(i=0; i<studentList.length ; i++) { // loop through each student element
   var stdNameVal, stdEmailList;
   studentNode = studentList.item(i);
   stdNameVal = studentNode.firstChild;
   stdEmailList = studentNode.getElementsByTagName("stdEmail");
   forStudent.innerHTML = forStudent.innerHTML  +
      stdNameVal.firstChild.nodeValue + "<BR>"
   for(temp=0; temp<stdEmailList.length; temp++){
      // loop through each e-mail belonging to the current student
      // forStudent is a <div> element in the <body> of the document
      forStudent.innerHTML = forStudent.innerHTML +
        "<A href=&quot; mailto:"  +
        stdEmailList.item(temp).firstChild.nodeValue +
        "&quot;>" + stdEmailList.item(temp).firstChild.nodeValue +
        "</A>" + "<BR>"
   }
} // for student list
```

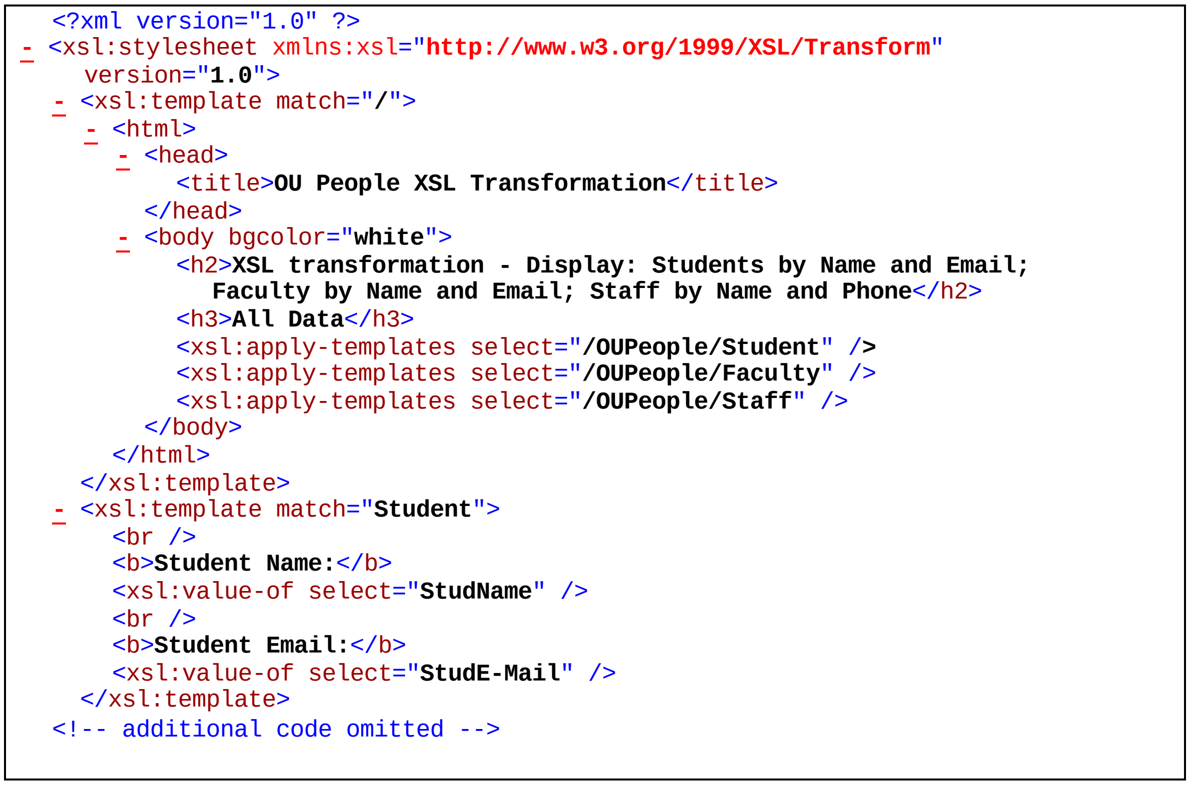
Figure 5. Sample JavaScript code for XML document manipulation.

The last major instructional section of the course dealt with the transformation of XML documents using Extensible Stylesheet Language Transformations (XSLT) and the XML Path Language (XPath). During this portion of the course students learned how to create and execute templates with iteration, sorting and conditional execution. Figure 6 shows part of an XSLT document created by a student, Quentin Pongrass, to transform OUpeople.xml into a list of people (students, then faculty, then staff) and their e-mail addresses.

The final project for the course asked the students to create an application that used programmatic control to manipulate XML documents. The assignment could be done using any combination of Java, JavaScript, and VBScript with Active Server Pages that the student chose. Projects completed by the students included a threaded discussion database (similar to a Lotus Domino application), a sample store (e-Commerce application), a contact management system, a test generation system and a course registration system.

CONCLUSIONS

The use of XML is growing in importance daily and appears to be a technology that will be around for some time. As such, it probably deserves a place in the curriculum of AACSB approved MIS programs. The course described in this paper demonstrates that undergraduate MIS students with a limited programming background are able to understand the concepts of XML and develop sample applications using XML technology, even within the constraints of a ten-week quarter.



```
  <?xml version="1.0" ?>
- <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
  - <xsl:template match="/">
    - <html>
      - <head>
          <title>OU People XSL Transformation</title>
        </head>
      - <body bgcolor="white">
          <h2>XSL transformation - Display: Students by Name and Email;
            Faculty by Name and Email; Staff by Name and Phone</h2>
          <h3>All Data</h3>
          <xsl:apply-templates select="/OUPeople/Student" />
          <xsl:apply-templates select="/OUPeople/Faculty" />
          <xsl:apply-templates select="/OUPeople/Staff" />
        </body>
      </html>
    </xsl:template>
  - <xsl:template match="Student">
      <br />
      <b>Student Name:</b>
      <xsl:value-of select="StudName" />
      <br />
      <b>Student Email:</b>
      <xsl:value-of select="StudE-Mail" />
    </xsl:template>
  <!-- additional code omitted -->
```

Figure 6. Portion of an XSLT document and the transformed page created by it.

REFERENCES

1 Abreu, Elinor (2001). XML Is Now the Standard Language for the Internet. Verified July 10, 2001 from http://www.thestandard.com/article/0,1902,24180,00.html.
2 Baller Pushes .NET, XML for Web Services (2001). Verified 10 Jul 2001 from http://www.thestandard.com/article/0,1902,27389,00.html.
3 Lawson, Stephen (2001). Web consortium adapts specification for XML definitions. Verified 10 July 2001 http://www.computerworld.com/cwi/story/0,1199,NAV47_STO60140,00.html.
4 XML Base and XLink Published as W3C Recommendations (2001) The XML Cover Page. Verified 10 July 2001 from http://xml.coverpages.org/ni2001-06-27-a.html.
5 A 'Rosetta Stone" for theWeb? The XML lingo could make it easier to find and use data (1999). Verified 10 July 2001 from http://www.businessweek.com:/1999/99_24/b3633183.htm.
6 Tromly, Maria (2000) Big names back new XML-based financial standard. Verified 10 July 2001 from http://www.computerworld.com/cwi/story/0,1199,NAV47_STO43799,00.html.
7 Tromly, Maria (2000) Big names back new XML-based financial standard. Verified 10 July 2001 from http://www.computerworld.com/cwi/story/0,1199,NAV47_STO46844,00.html.
8 Meehan, Michael (2000) Airlines turn to XML to try to fix e-ticket transfer problems. Verified 10 July 2001 from http://www.computerworld.com/cwi/story/0,1199,NAV47_STO51231,00.html.
9 Sliwa, Carol (2001) XML-based retail applications moving closer to reality Verified 10 July 2001 from http://www.computerworld.com/cwi/story/0,1199,NAV47_STO56013,00.html.
10 Weiss, Todd (2001) XML group to create specifications for voting system. Verified 10 July 2001 from http://www.computerworld.com/cwi/story/0,1199,NAV47_STO60214,00.html.
11 Deitel, Deitel, Nieto, Lin & Sadhu (2001) *XML How To Program*, Prentice Hall.