Distance Education Using XML Technology

Senay Kafkas and Zeki Bayram
Department of Computer Engineering,
Eastern Mediterranean University,
Famagusta, Turkish Republic of Northern Cyprus
{senay.kafkas, zeki.bayram}@emu.edu.tr

***Abstract*-Distance Education is an instructional delivery system that connects learners with educational resources. In this paper, we report a system that realizes distance education by using XML technology. The system provides a WEB interface to students who get course materials and take tests online using their browser. At the same time, it provides tools for instructors to design course materials and exams on their local machine and upload the materials to the server as XML documents.**

I. INTRODUCTION

Distance education (DE) is an instructional delivery system that connects learners with educational resources. DE can take place when instructor(s) and learner(s) are physically separated from one other. With the widespread use of the Internet and the World Wide Web (WWW), there has been a lot of interest in using this new medium for DE [1,3,5]. Especially the rapid growth of the media-rich extensions of the WWW allows new developments in the way instructors transfer knowledge to their students [2].

A wide range of technological options is available to the distant educator. These options fall into four major categories: voice, video, print and data. Making the appropriate choice of the technology to be utilized in DE is important for the success of the learning process [4]. In this paper we report on a DE project that is based on the data category that makes use of XML and related WEB technologies.

"Distance Education Using XML Technology" connects students with teachers via the Internet. It provides a WEB interface to students who get course materials and take tests online using their browser. At the same time, it provides tools for instructors to design course materials and exams on their local machine and upload the materials to the server as XML documents. In fact, all data is both stored and transmitted in XML format. Proper authentication and authorization is enforced for both instructors and students.

The rest of the paper is organized as follows. Section II describes the overall system model. Section III gives a detailed description of the XML files and their schemas used in the system. Sections IV and V provide detailed information about the client-side application and server-side web application respectively. Finally, section VI concludes the paper.

II. THE SYSTEM MODEL

Figure 1 illustrates the overall system model. In the model, there is a WEB server, which using ASP technology uploads course materials and tests from instructor PC's in the form of XML documents. At the same time, students access the online courses through their standard browsers and the HTTP protocol, with no additional software being required.

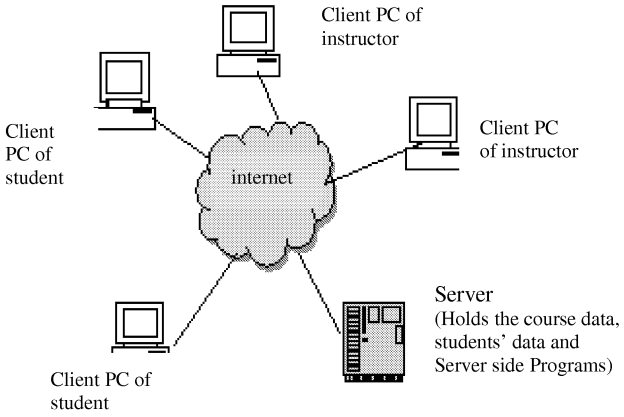

Figure 1: System Model

The server in the system is used as a "Domain Instruction Server" that is a repository consisting of web deliverable instruction units [6]. It holds server-side programs as well as XML files that contain course and examination data. The processing of XML files is done through the Microsoft XML parser (MSXML) and the Document Object Model (DOM), using ASP WEB technology and the VBScript programming language. Each distant course is represented by two XML files on the server. One of these holds the course data, and the other online examination data of the course in the form of multiple choice questions. The data in the course and examination files are extracted and presented to the student as a dynamically constructed web page. Server-side programs also accept requests for uploading XML data from clients (i.e. computers of instructors), and store the course files in their respective locations on the server's hard disk.

The client-side program that is used on the instructors' computers for designing and uploading course content is implemented using Visual Basic technology. Instructors use this program to generate, maintain and upload to the server XML files for the courses. Again, the MSXML parser and DOM are used for this purpose. The upload operation of XML files is done through the HTTP protocol.

III. XML FILES USED IN THE SYSTEM

Figure 2 illustrates tasks involved in parsing an XML document [7]. The XML parser takes an existing XML document and generates its in-memory representation in the form of a tree which is made available to script or application code through the document object model (DOM). Once the in-memory processing is finished, the application code or script can serialize the tree to a physical XML file through the *save* method. Completely new XML documents can also be created in memory using DOM, and serialized to files.

In our project, we make extensive use of the MSXML parser for processing XML documents.

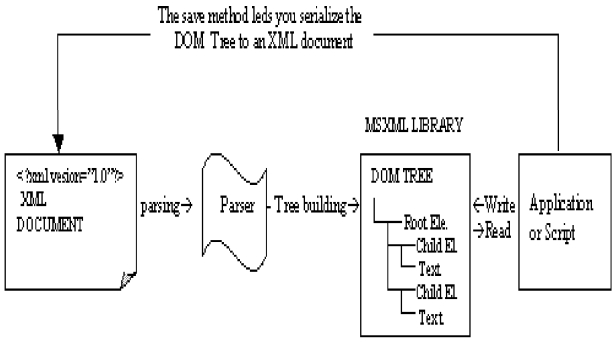

Figure 2: Parsing XML documents

On the WEB server separate XML files represent each course's lecture notes and its examination. An example lecture notes file is given in figure 3 and a sample examination is given in figure 4.

```
<Course>
  <Course_id>Cmpe418</Course_id>
    <AllLectures>
      <Lecture>
          <lname>XML</lname>
          <sublecture>
              <sname>what is xml?</sname>
               <content>
                    The buzz-word "XML" is      beginning
                    to pop up all over the Net, and in the
                    Goodies e-mail box.....
               </content>
           </sublecture>
           <sublecture>
              <sname>dtd</sname>
              <content>SGML is what is known as a
                          "meta-language;" it allows a
                          programmer to write a DTD
                          (Document Type Definition) that
                          numerous pages can follow...
              </content>
          </sublecture>
      </Lecture>
    </AllLectures>
</Course>
```

Figure 3:Sample Lecture notes file "Cmpe418.xml"

In figure 3, the root element is C*ourse.* On the second level, there are the elements *Course_id* that holds the id of a course and *AllLectures* that contains *Lecture* elements. Each *Lecture* element contains information about one full lecture. It consists of *lname* and *sublecture* elements. The *lname* element holds the name of the lecture. These nodes are main topics of lectures (for example, HTML, XML etc.). *Sublecture* elements contain subparts of the lecture.

Each course has an online-examination part. The figure 4 depicts a sample examination file. In this figure, *exam* is the root element. Under the root, there are *question* nodes. Each *question* node represents a question. It has *num* node to hold the question number, *que* node to hold the actual question, *a, b, c, d* nodes to hold four different answers and *correct* to hold the correct answer of the question.

```
<exam>
     <question>
       <num>1</num>
       <que>I hope you .....the election to be held  soon.</que>
       <a>are winning</a>
       <b>will win</b>
       <c>have won</c>
       <d>are going to win</d>
       <correct>b</correct>
     </question>

    <question>
       <num>2</num>
       <que>It ........sunny tomorrow but just know  it.....</que>
       <a>will be/is raining</a>
       <b>is / will rain</b>
       <c>is going to be/ rains</c>
       <d>is/rains</d>
       <correct>a</correct>
     </question>

   <question>
       <num>3</num>
       <que>Jason..... join the army next month</que>
       <a>will</a>
       <b>going to</b>
       <c>may</c>
       <d>had to</d>
       <correct>b</correct>
     </question>
</exam>
```

Figure 4: Sample examination file "Exam.xml"

Figures 5 and 6 depict the schema of course and examination files respectively. The XML files used in the system were validated by using the validator at [8].

In figure 5, the root is complex type *Course* element. It contains the simple type *Course_id* and the complex type *AllLectures* elements. *Course_id*'s type is *string* and it must appear exactly once in the XML document. The type of *AllLecture's* is *ALec_Type.* Again this element must appear just once. *AllLectures* contains complex type *Ltype* and an unbounded number of *Lecture* elements. *Each Lecture* element contains simple type (string) *lname* and complex type

(*slType*) *sublecture* elements. The *sublecture* element has two simple type (string) elements, *sname* and *content.*

```
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Course">
  <xsd:complexType>
  <xsd:sequence>
<xsd:element name="Course_id" type="xsd:string" minOc
curs="1" maxOccurs="1"/>
<xsd:element name="AllLectures" type="ALec_Type" mi-
nOccurs="1" maxOcurs="1"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:complexType name="ALec_Type">
<xsd:sequence>
<xsd:element name="Lecture" type="LType"
minOccurs="0"  maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="LType">
<xsd:sequence><xsd:element name="lname"
type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="sublecture" type="slType"
minOcurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="slType">
<xsd:sequence>
<xsd:element name="sname" type="xsd:string"/>
<xsd:element name="content" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Figure 5: Schema of lecture notes

```
<xsd:schema
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="exam">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="question" type="QType" minOc-
curs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:complexType name="QType">
<xsd:sequence>
<xsd:element name="num" type="xsd:integer" minOc-
curs="1" maxOccurs="1"/>
<xsd:element name="que" type="string" minOccurs="1"
maxOccurs="1"/>
<xsd:element name="a" type="xsd:string" minOccurs="1"
maxOccurs="1"/>
<xsd:element name="b" type="xsd:string" minOccurs="1"
maxOccurs="1"/>
<xsd:element name="c" type="xsd:string" minOccurs="1"
maxOccurs="1"/>
<xsd:element name="d" type="xsd:string" minOccurs="1"
maxOccurs="1"/>
<xsd:element name="correct" type="xsd:string" minOc-
curs="1" maxOccurs="1"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Figure 6: Schema of online examination

In figure 6, the root element is *exam* that contains complex type (*QType*) *question* elements. Each *question* element contains simple type (integer) *num* that is question number, simple type (string) *a, b, c, d* that are four different answer choices and simple type (string) *correct* that is correct answer of the question.

IV. THE CLIENT-SIDE APPLICATION

Instructors use their client-side program to maintain the course lectures and examination files that reside on the server. They can add, delete or change a main lecture topic (*lname*), a sub lecture topic (*sname*) or content (*content*). A password scheme makes sure that only instructors who have an online-course can update the files.

Figure 7 is a screenshot from the client-side program that shows how the instructors can add a sub lecture and its content.

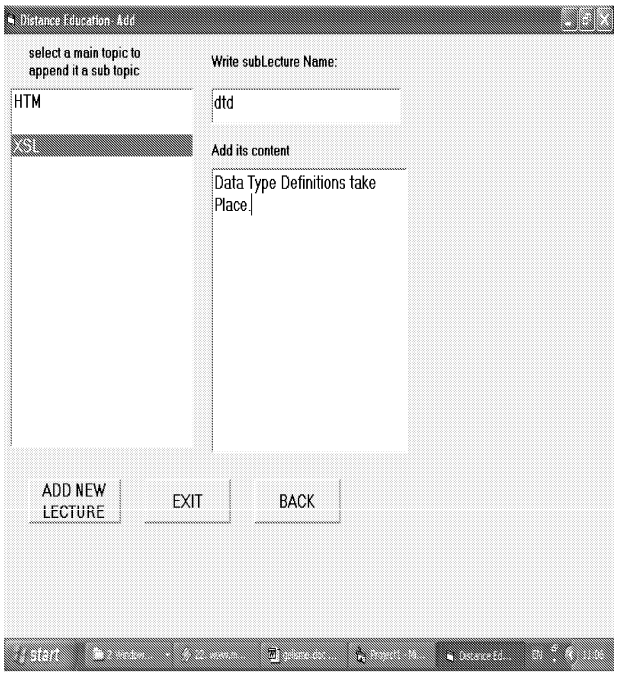

Figure 7: Adding new sub topic and its content.

Figure 8 is a code segment of client-side program. In the code, *xmlDoc1* is defined as Msxml2.DOMDocument object. It loads the related course file that we wish to maintain. Variables of type IXMLDOMNode and IXMLDOMelement are used to reach the node and root elements of the XML file respectively. The *xmlhttp* variable is defined as a Msxml2.xmlhttp object and used to "POST" the XML file by HTTP protocol to an ASP page. This ASP page (servercmpe418.asp) in figure 9, runs on the server and saves the posted XML document.

```
.....

Dim xmlhttp As New Msxml2.xmlhttp
Dim cNode, newNode, newNode2 As IXMLDOMNode
Dim  newNode3   As IXMLDOMNode
Dim xmlDoc1 As New  Msxml2.DOMDocument40
Dim root, root2 As IXMLDOMElement
xmlDoc1.async = False

xx = "http://localhost/extra/"
yy = ".xml"
st = xx + ccode + yy
xmlDoc1.Load (st)
x2 = "http://localhost/extra/server"
y2 = ".asp"
str2 = x2 + ccode + y2

xmlhttp.open "POST", str2, False
.....
```

Figure 8: A code segment of client-side programming

V. THE SERVER-SIDE WEB APPLICATION

The server holds ASP pages as well as XML files that make Distance Education Web Site. The Distance Education Web Site is designed for students who take one of those online courses. By using this web site students can reach the lecture notes of their courses or they can take online examinations. Registered students' information are kept in a database so proper authentication is enforced for those students.

Figure 9 shows part of the code of a server side program. In the code, *xmldoc2* is assigned an Msxml.DomDocument object. It loads the request that posted from client-side program and saves the modified XML document on the server.

```
<%  dim xmldoc2
set xmldoc2 =
Server.CreateObject("Msxml2.DOMDocument.4.0")

xmldoc2.async = false
xmldoc2.load(Request)
xmldoc2.save(Server.MapPath("cmpe418.xml"))%>
```

Figure 9: servercmpe418.asp

Figure 10 and figure 11 are lecture notes and online-examination screenshots respectively from the distance education web site. Students can view lecture notes of the online courses that they are registered to. Furthermore, students can use this web site to take their online examination. The question format of the examination is multiple choice and the duration is restricted. Whenever the duration expires the questions' page automatically redirects the student to the result page.

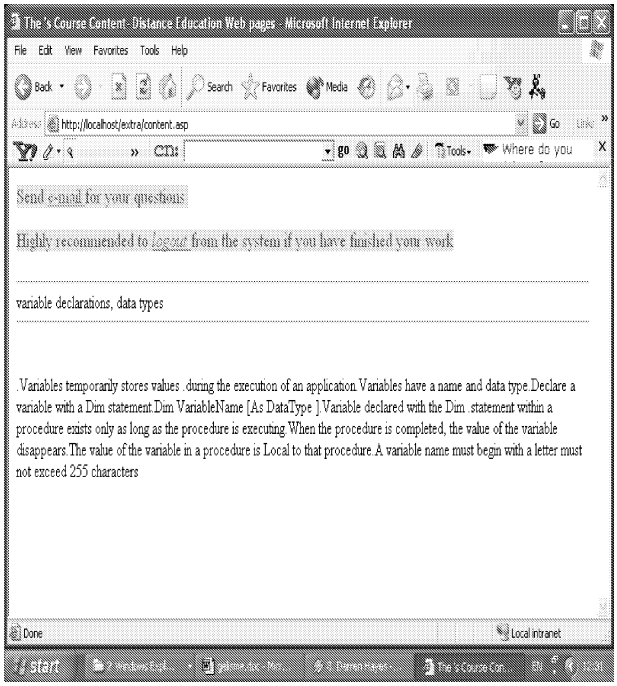

Figure 10: Lecture Notes

Figure 11 is an online examination screenshot. A student can take a given examination only once.

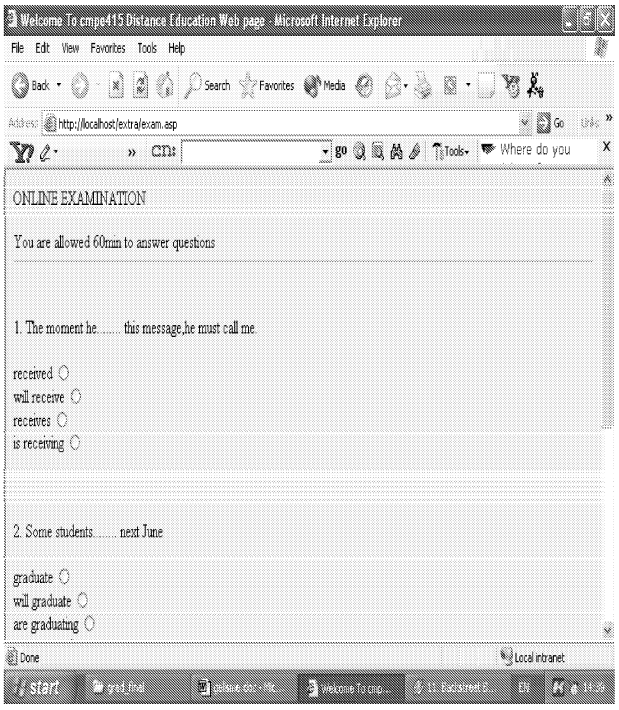

Figure 11: Online-examination

ASP pages process XML files through the MSXML parser and the Document Object Model (DOM). Figure 12 is a code segment that runs behind the web page in Figure 11. In the code the XML file of examination is parsed and question nodes are printed to the students.

```
<html><head><title>Welcome To  <% response.write
(session("courses"))%> Distance Education Web age</title>
<body bgcolor="#FFFFCC">
<form method="post" action="result.asp">
<% Dim xmlFile, xmlNodes, xmlItem
Dim strPath, strTitle, strFileName
Dim j, currNode, currNode2, num, i,k
Dim nums,ques,aa,bb,cc,dd
.....

<%Set xmlFile = Server.CreateObject( "Microsoft.XMLDOM"
)
xmlFile.Async = False
xmlFile.Load( strPath )
Set xmlNodes = xmlFile.getElementsByTagName("question")
Set nums = xmlFile.getElementsByTagName("num")
Set ques = xmlFile.getElementsByTagName("que")
Set aa = xmlFile.getElementsByTagName("a")
Set bb = xmlFile.getElementsByTagName("b")
Set cc = xmlFile.getElementsByTagName("c")
Set dd = xmlFile.getElementsByTagName("d")
k=0
For j = 0 To (xmlNodes.length - 1)
Response.write (j + 1 & ".   " & ques.Item(j).Text) %>
<br/><br/><label>
   <% Response.write(aa.Item(j).Text)  %>
<input name="<%Response.write(nums.Item(j).Text) %>"
type="radio" value= "a"/ >
</label><br/>
<label>
<% Response.write(bb.Item(j).Text)  %>
<input name="<%Response.write(nums.Item(j).Text) %>"
type="radio" value= "b"/ >
</label> <br/>
  <label>
   <% Response.write(cc.Item(j).Text)   %>
<input name="<%Response.write(nums.Item(j).Text) %>"
type="radio" value= "c"/ >
</label> <br/>
<label>
   <% Response.write(dd.Item(j).Text %>
<input name="<%Response.write(nums.Item(j).Text) %>"
type="radio" value= "d"/ >
</label><br/><br/><br/><br/>
<% Next  %>
<br/><br/>
<input type="submit" value="Press to learn your result"/>
</form></body></html>
```

Figure 12: A code segment of server-side programs

VI. Conclusion And Future Work

In this paper, we report a system (Distance Education Using XML Technology) that connects instructors with students via the Internet. This system provides tools for instructors to maintain and update their course materials and exams from their local machines. All the course materials and exams are saved as XML files on a server. Students access the course materials and exams through a standard WEB browser interface.

In the design of the system, Visual Basic technology was used for the client-side programming and ASP was used for for the server-side programming. Our plans for the future include porting the system to the .NET platform, using WEB services and enhancing it with tutoring features.

References

[1] P. Brusilovsky et alt., "Web-based education for all: a tool for development adaptive courseware," Proceedings of seventh international world wide web conference, April 98, Brisbane Australia.

[2] Latchman, H, Salzmann, C, Gillet, D, Bouzekri, H, "Information Technology Enhanced Learning in Distance and Conventional Education," IEEE Trans. Education, Vol. 42, No. 4, Nov. 1999, pp.247-254.

[3] Chandler, J, Fontenot, D, Hagler, M, Marcy, W, "Why the Distinction between On-Campus and Dis tance Learning is Blurring," 29 Education Conference (CD-ROM), Nov. 10 - 13 1999, San Juan, Puerto Rico, pp.12a2.11-12a.15.

[4] M. Liamara, S. Comassetto, "SEEADE- A Model of Decisions Choices about Technologies in the Distance Education Based on Politic-pedagogic Projects," International Conference on Engineering Education 2002, 18-22 August 2002, UMIST, Manchester, UK.

[5] C. Lin et al, "The Research of Constructing the Environment of Web-Based Cooperative learning community", International Conference on Engineering Education 2002, 18-22 August 2002, UMIST, Manchester, UK.

[6] Juan E. Gilbert, Dale-Marie Wilson, "Domain Instruction Server (DIS) ", Proceedings of the IEEE Conference on Advanced Learning Techniques," 2001.

[7] MSXML 4.0 Parser SDK Package.

[8] http://apps.gotdotnet.com/xmltools/xsdvalidator/Default.aspx