December 2001

Mapping UML to XML: An Object-Oriented Approach to Developing E-Business Applications

Jack Marchewka
Northern Illinois University

Chang Liu
Northern Illinois University

MAPPING UML TO XML: AN OBJECT-ORIENTED APPROACH TO DEVELOPING E-BUSINESS APPLICATIONS

Jack T. Marchewka
Northern Illinois University
jtm@niu.edu

Chang Liu
Northern Illinois University
cliu@niu.edu

Abstract

XML (eXtensible Markup Language) has been gaining notoriety and interest in electronic business applications. Although several XML development tools are currently available, each has its own strengths and weaknesses in terms of ease of use and conceptual foundation. Often novice XML developers focus on the XML code; however, this approach can result in more rework, errors, and inefficiencies. Using a conceptual model for development and design may provide a better real world abstraction and improved communication and understanding between the developer and user. In this paper, we show how the UML class diagram can seamlessly map to an XML-based electronic business application.

eXtensible Markup Language (XML)

XML is a relatively new technology that has been gaining a great deal of attention. The power of XML is its potential to separate data from definition, whereby the developer has the freedom to build an application with its own data structure. By providing a common language for describing data, XML will enable more precise searching, let businesses share data more efficiently, and make navigating data much easier (Gottesman 1998). This gives organizations the ability to not only use the same data definitions across different platforms, but allows different organizations to use the same data differently. In addition, XML is based on UNICODE. This allows for XML to support not only the Western characters supported by ASCII, but Asian characters as well. As a result, XML has been of particular interest to organizations conducting electronic commerce globally.

XML is a subset of the Standard Generalized Markup Language (SGML) and is therefore supported by defined by the World Wide Web Consortium (W3C at www.w3c.org). Similar to its cousin, HTML, XML uses tags that are parsed by a Web browser. However, unlike HTML, XML allows each application to define how the data will be defined and what will be done with it. In short, XML focuses on the content, while HTML focuses on the presentation.

At the core of XML is a Document Type Definition (DTD) or a schema used to define the data contents of the XML document. In addition, an XML document must be "well-formed" and "valid." A well-formed XML document conforms to the syntax rules set up for XML by the W3C in the XML 1.0 specification. However, this really means that the XML document contains one or more elements, and one element, called the root, contains all the other elements. In addition, a well-formed XML document means each element also nests inside any enclosing elements properly. Several tools are now available on the market to facilitate the development of the DTD or schema; however, each has its own inherent strengths and weaknesses in terms of ease of use and conceptual foundation.

On the other hand, an XML document is "valid" if it references a DTD or schema, and the DTD or schema complies with the W3C standard. Although most XML browsers will check to ensure that the XML document is well formed, only a few have the capability to check to verify that it is valid.

Unified Modeling Language (UML)

The Unified Modeling Language (UML) is a product of object-oriented analysis and design (OOA & D) that first appeared in the late 1980s and early 1990s and unifies the methods proposed by Booch, Rumbaugh, and Jacobson (Fowler and Sott 1997). OOA & D is a direct result of object-oriented programming languages such as SmallTalk and C++. However, OOA & D is not limited to designing application systems developed in object-oriented programming languages.

As a modeling language, UML provides the graphical notation and syntax used to express a design. Moreover, as a modeling language, UML provides a key technique for improving communication between developers and users and provides a common vocabulary of object-based terms and diagramming techniques rich enough to model an applications system from analysis through design (Dennis and Haley 2000).

An object-oriented approach to systems analysis and design has several benefits over the traditional process or data focused approaches. More specifically, information systems developed using these traditional approaches have been regarded as being more error-prone, expensive to build, and inflexible to maintain (Satzinger and Orvik 2001). Since each object is small and self-contained, the object-oriented approach is more manageable since the complexity of systems development is reduced. In addition, the object-oriented approach provides a higher degree of reusability. The potential benefits of reusability include a higher quality system that is less expensive to build and maintain (Satzinger and Orvik 2001).

The Class Diagram

Objects are the heart of the object-oriented approach, since just about everything can be viewed as an object. UML defines a set of nine object-oriented diagramming techniques that can be used to model a system (Dennis and Haley 2000). One of the key diagramming techniques is the class diagram. The class diagram describes the types of objects (i.e., classes) in a system and the static relationships that exist among them. Relationships can be associations (e.g., a customer orders a product) or subclasses (a customer is a kind of person). In addition, class diagrams define the operations or methods (i.e., things an object or class can do), as well as the properties or attributes of a particular class.

In addition to associations, class diagrams can be useful for defining aggregation and composition. More specifically, aggregation allows the developer to model is-a-part-of relationships (e.g., a wing is a part of an airplane). By modeling object classes and their various relationships, the developer is able to create a better abstraction of the real world system. As a result, an application system can be developed that better reflects the needs of the user or organization.

Mapping UML to XML Documents

Although XML has been gaining interest and popularity, the design of a valid and well-formed XML document and XML-based e-business application remains a challenge. It is important to decide how the data content should be represented to support device independence, searchability, and efficient Internet data interchange. Therefore, an XML development process should be carried out in a rational, disciplined, effective, controlled, and uniform way to design e-business applications. Even though XML is not an object-oriented language, the opportunity to apply OOA& D techniques exists. The purpose of this paper is to show how UML can be used to map to XML in order to create an electronic business application.

Figure 1 provides an example of a class diagram for an application used to create a purchase order used by a large, Midwestern university.

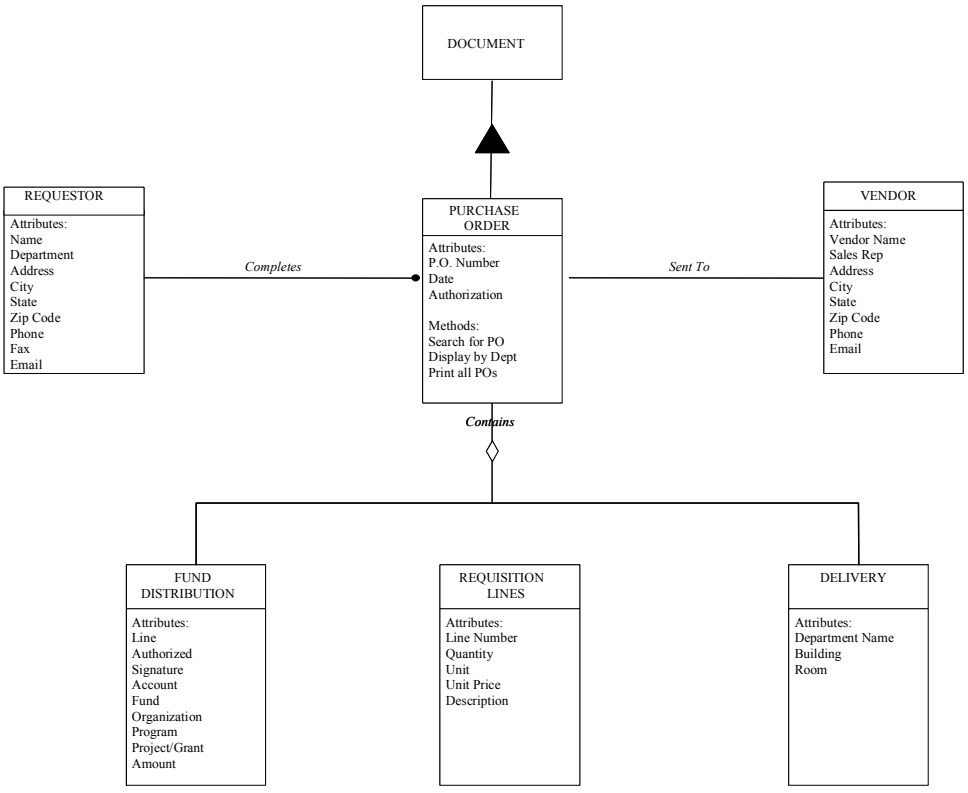

Figure 1. Class Diagram of a Purchase Order

Mapping the DTD

As illustrated in Figure 1, the class diagram shows the classes, attributes, and various associations. It provides a guideline for the design of the XML DTD. For example, the super class DOCUMENT can be used as a root element called XML_DOCUMENT. Other subclasses in Figure 1 can be mapped to a grouping level within the XML DTD. In addition, the attributes in the class diagram can be mapped as data points associated with their grouping levels. Figure 2 illustrates how the class diagram maps to the DTD structure.

```
<!ELEMENT XML_DOCUMENT (PURCHASE_ORDER, REQUESTOR, VENDOR, FUND+,REQUISTIONLINE+, DELIVERY)>

<!ELEMENT PURCHASE_ORDER (PONumber, Date, REQUESTOR, VENDOR, FUND+, REQUISTIONLINE+,DELIVERY)+>
<!ELEMENT PONumber (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT REQUESTOR (Name, Department, Address, City,State, Zip_Code, Phone, Fax, Email)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Department (#PCDATA)>
<!ELEMENT Address (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT State (#PCDATA)>
<!ELEMENT Zip_Code (#PCDATA)>
<!ELEMENT Phone (#PCDATA)>
<!ELEMENT Fax (#PCDATA)>
<!ELEMENT Email (#PCDATA)>

<!ELEMENT VENDOR (VendorName, SalesRepresentative, Address, City, State, Zip, Phone, Email)>
<!ELEMENT VendorName (#PCDATA)>
<!ELEMENT SalesRepresentative (#PCDATA)>
<!ELEMENT Address (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT State (#PCDATA)>
<!ELEMENT Zip (#PCDATA)>
<!ELEMENT Phone (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
```

```
<!ELEMENT FUND (Line, Authorized_Signature, Account, Fund, Organization, Program, Project, Amount)>
<!ELEMENT Line (#PCDATA)>
<!ELEMENT Authorized_Signature(#PCDATA)>
<!ELEMENT Account (#PCDATA)>
<!ELEMENT Fund (#PCDATA)>
<!ELEMENT Organization (#PCDATA)>
<!ELEMENT Program (#PCDATA)>
<!ELEMENT Project (#PCDATA)>
<!ELEMENT Amount (#PCDATA)>

<!ELEMENT REQUISTIONLINE (LineNumber, Quantity, Unit, UnitPrice, Description)>
<!ELEMENT LineNumber (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT Unit (#PCDATA)>
<!ELEMENT UnitPrice (#PCDATA)>
<!ELEMENT Description (#PCDATA)>

<!ELEMENT DELIVERY (DepartmentName, Building, Room)>
<!ELEMENT DepartmentName (#PCDATA)>
<!ELEMENT Building (#PCDATA)>
<!ELEMENT Room (#PCDATA)>
```

Figure 2. Purchase Order DTD Mapped to the Class Diagram Using Elements

As illustrated in Figure 2, XML elements represent classes and the sub elements represent the attributes from the class diagram in Figure 1. Although we chose to use elements in our XML document, XML attributes could have been used and would map just as well.

Generating the XML Document Based on the DTD

Once the DTD is created, it can be referenced either externally or internally from the XML document. Figure 3 illustrates some of the code for the XML document with an external reference to the DTD. This results in a valid XML Document.

```
<?xml version="1.0" ?>
<!DOCTYPE XML_DOCUMENT SYSTEM “PurchaseDTD.dtd” >
<XML_DOCUMENT>
    <PURCHASE_ORDER>
        <PONumber>1001</PONumber>
        <Date>3/3/2001</Date>
        <Authorization>Joe Simth</Authorization>
        <REQUESTOR>
            <Name>Robert Rollins</Name>
            <Department>Information Systems</Department>
            <Address>100 King Street</Address>
            <City>Dekalb</City>
            <State>IL</State>
            <Zip_Code>60115</Zip_Code>
            <Phone>815-753-1285</Phone>
            <Fax>815-753-1286</Fax>
            <Email>rrollins@niu.edu</Email>
        </REQUESTOR>
        <VENDOR>
            <VendorName>Micro Technology</VendorName>
            <SalesRepresentative>Mary Jones</SalesRepresentative>
            <Address>2000 First Street</Address>
            <City>Dekalb</City>
            <State>IL</State>
            <Zip>60115</Zip>
            <Phone>815-758-8888</Phone>
            <Email>maryjones@micro.com</Email>
        </VENDOR>
        <FUND>
            <Line>2</Line>
            <Authorized_Signature>Hobert Ruth</Authorized_Signature>
            <Account>31</Account>
            <Fund>Small Business</Fund>
            <Organization>College of Business</Organization>
            <Program>MBA</Program>
            <Project>Distance Learning</Project>
            <Amount>2000</Amount>
        </FUND>
        <REQUISTIONLINE>
            <LineNumber>101</LineNumber>
            <Quantity>2</Quantity>
            <Unit>Package</Unit>
            <UnitPrice>$999</UnitPrice>
            <Description>Microsoft Commerce Server 2000</Description>
        </REQUISTIONLINE>
        <DELIVERY>
            <DepartmentName>Information Systems</DepartmentName>
            <Building>Wirtz Hall</Building>
            <Room>209</Room>
        </DELIVERY>
    </PURCHASE_ORDER>
</XML_DOCUMENT>
```

Figure 3. The XML Document Based on the DTD

Mapping Object Methods

Although XML provides structure for the content of the data, it does not specify how the data is to be presented. However, the methods defined in the class diagram can be represented using the XML Document Object Model (DOM). The DOM consists of a set of programming objects that represent the different components of an XML document. The methods of these objects allow the use of scripts to display the XML document from within an HTML page (Young 2000).

Methods can be classified into three types: (1) constructor, (2) query, and (3) update. A constructor method creates a new instance of a class; a query method searches the instance of an object; and an update method can alter the state of an object. For example, JavaScript can be used to create, search, and update the XML-based purchase order. In addition, HTML can be used to format the contents of the document, while XML is used to define the structure of the data. Figure 4 illustrates how the methods defined in the class diagram map to the XML document and HTML document.

As an illustration, we demonstrate the concept in Figure 4 to display all purchase orders and to search for a specific purchase order. Figure 5 shows partial code of information presentation and Figure 6 shows the search for a specific purchase order.

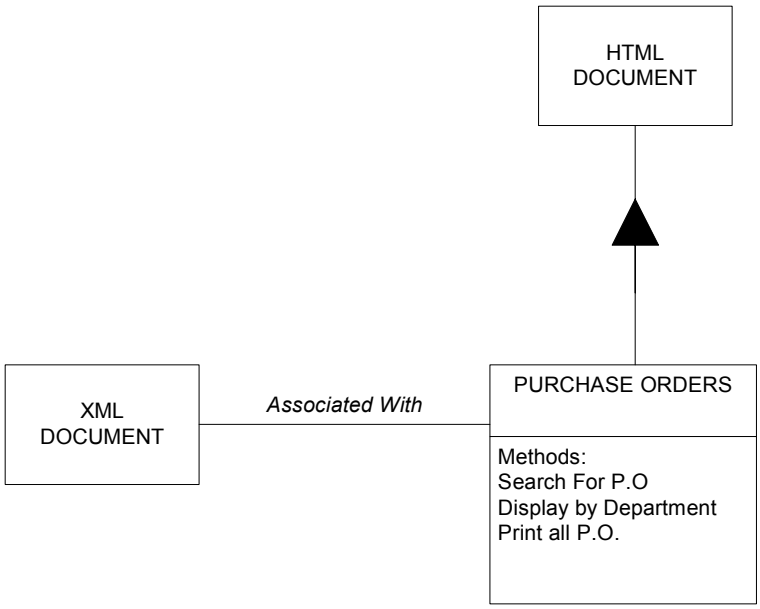

Figure 4. Mapping UML Method to Generate HTML Output

```
<html><head><title>Purchase Description</title>
<SCRIPT LANGUAGE="JAVASCRIPT" FOR="window" EVENT="ONLOAD">
 Document=dsoPurchase.XMLDocument;
 HTMLCode="";
 for (i=0; i<Document.documentElement.childNodes.length; i++)
 {
    HTMLCode+="<table border=0><tr><th><i><u>P.O. Number: </i></u></th>"
      +"<th><i><u>Date: </i></u></th>"
      +"<th><i><u>Authorization: </i></u></th></tr>"
      +"<tr><th>"+Document.documentElement.childNodes(i).childNodes(0).text+"</th>"
      +"<th>"+Document.documentElement.childNodes(i).childNodes(1).text+"</th>"
      +"<th>"+Document.documentElement.childNodes(i).childNodes(2).text+"</th>"
      +"</tr></table>"
  }
         DisplayDIV.innerHTML=HTMLCode;
</Script>
</head>
<body>
    <XML ID="dsoPurchase" SRC="Purchase2.xml"></XML>
    <h2>Purchase Description</h2>
    <DIV ID="DisplayDIV"></DIV>
</body></html>
```

Figure 5. Mapping Class Diagram Method to Generate HTML Output

Purchase Description

Please enter a P.O. Number 1001

P.O. Number:	Date:	Authorization:
1001	3/3/2001	Joe Simth

Requestor

Name:	Department:	Address:	City:	State:	Zip:	Phone:	Fax:	Email:
Robert Rollins	Information Systems	100 King Street	Dekalb	IL	60115	815-753-1285	815-753-1286	rrollins@niu.edu

Vendor

Vendor Name:	Sales Repl::	Address:	City:	State:	Zip:	Phone:	Email:
Micro Technology	Mary Jones	2000 First Street	Dekalb	IL	60115	815-758-8888	maryjones@micro.com

Fund

Line:	Authorized:	Account:	Fund:	Organization:	Program:	Project:	Amount:
1	Gary Smith	33	BITTC	College of Business	Management Information Systems	B2B Commerce Application	1500

Line Item

Line Number:	Quantity:	Unit:	Unit Price:	Description:
100	20	Package	$99	Microsoft Visual InterDev 6.0
101	2	Package	$999	Microsoft Commerce Server 2000
102	30	Package	$69	Microsoft Visual Basic 7.0

Figure 6. Sample Output for Purchase Order Search

Conclusion

Developing electronic business applications using XML requires tools, methods and discipline to design an application that meets the needs of the organization. Starting from the code, results in wasted time, energy, and resources; however, using a design method such as UML can result in better communication and understanding of the business requirements. In this paper, we showed how UML may map seamlessly to XML in order to develop a well formed and valid document to support electronic business.

Although XML has been gaining interest and popularity, the design of a valid and well-formed XML document and XML-based e-business application remains a challenge. This challenge is further complicated by the research question: how should XML be effectively extended to support the representation of business information (Smith and Poulter, 1999)? It is important to decide how the data content should be represented to support device independence, search-ability, and efficient Internet data interchange. Therefore, an XML development process should be carried out in a rational, disciplined, effective, controlled, and uniform way to design e-business applications. Even though XML is not an object-oriented language, the opportunity to apply OOA& D techniques exists.

This paper provides a basic introduction for designing XML-based applications. Future research should focus on larger scale applications and how an object-oriented systems analysis and design approach can support this development effort. At the present time, the Object Management Group (OMG) gas proposed XMI (XML Metadata Interchange) as an open model that identifies standard XML DTDs for transferring and exchanging UML models and MOF meta models over the Internet.

References

Dennis, A. and W. B. Haley (2000). *Systems Analysis and Design: An Applied Approach.* New York, NY, John Wiley & Sons, Inc.

Fowler, M. and K. Sott (1997). *UML Distilled: Applying the Standard Object Modeling Language.* Reading, MA, Addison-Wesley.

Gottesman, B. Z. (1998). "Why XML Matters." *P.C. Magazine* (October 6): 217-238.

Satzinger, J. W. and T. V. Orvik (2001). *The Object-Oriented Approach: Concepts, System Development, and Modeling with UML.* Boston, MA, Course Technology.

Smith, H. and Poulter, K. (1999). Share the ontology in XML-based trading architectures, *Communications of the ACM*, 42(3), pp. 110-111.

Young, M. (2000). *Step By Step XML,* Microsoft Press.