COPYRIGHT Notice

This paper was presented at CUMREC 2000, the Higher Education Administrative Technology conference. It is the intellectual property of the author(s). Permission to print out or disseminate all or part of this material is granted provided that the copies are not made or distributed for commercial advantage and that the title and authors of the paper appear. To copy or disseminate otherwise, or to republish in any form, requires written permission from the authors.
XML and Student Information Systems

Author
Leo Fernig, Project Manager, Systems Group, Student Services (lfernig@regi.ubc.ca)

Institution
University of British Columbia (www.ubc.ca) in Vancouver, British Columbia is a full research oriented university. It is the 3rd largest university in Canada. Twelve faculties and ten schools offer a combination of over 85 undergraduate, post-undergraduate and graduate degrees, 20 diploma and 4 certificate programs. There are currently over 35,000 students.

Abstract
Beneath all the hype that surrounds XML there are some very serious trends. At UBC we are beginning to use XML for some parts of our internet based Student Information Systems. This paper is a high level examination of the implications of an “XML aware” student information system. It contains brief overviews of XML and current XML technologies. It then examines how XML fits into the overall architecture of an Internet based Student Information System.
XML and Student Information Systems

XML is the subject of an enormous amount of hype. Articles on XML display headlines like “XML: The Lifeblood of E-Commerce”. Yet, beneath all the hype, there are some very important trends that need to be understood by anyone who is involved in the development of information systems. XML is transforming technologies throughout the industry. Oracle Corporation has built XML support directly into their database products. The same applies to IBM’s relational database product. Microsoft’s Internet Explorer 5.0 has extensive XML support. These are just a few well-known commercial products that have become “XML aware”. There are also numerous initiatives to use XML for EDI transactions (both for commercial EDI and education). The development and refinement of XML lies at the center of the efforts of the World Wide Web Consortium. XML may very well become the standard way the semantics of common business objects are captured and passed across a network from one place to another. Traditional EDI has tended to be the preserve of large corporations. The reason that XML is causing so much excitement is the expectation that the new technologies will be cheaper and more widespread thus allowing the mass of small and medium businesses to engage in e-commerce.

At UBC we have spent the last two years redeveloping our student information systems. We have adopted an end-to-end Java solution and we deliver everything over the Internet. We have reached a stage where we are quite satisfied with our core architecture. We now want to start opening up our services in a way that will give students a more seamless access to everything they need on campus. For instance, we are building automatic feeds between the SIS and internet based teaching tools. Eventually we want to allow students to use the same portal for registration as they do for course materials. In order to make this architecture work we are beginning to publish some of our core transactions sets (such as authentication requests, class rosters, course schedules and course descriptions) as XML documents. We also create an XML extract from our curriculum database and this is the main input to our publications software for the University Calendar. Generally speaking, we are approaching the issue of making our student information systems “XML aware” in a piecemeal and empirical manner. What we really need is to approach the problem in a more systematic fashion. That is the purpose of this paper. It is intended as an examination of what is involved in creating an “XML aware” student information system in the wider context of the various synergies that have been generated by XML technologies. It consists of four sections:

1. a brief overview of XML.
2. a description of different commercial XML technologies.
3. an overview of what the architecture of an XML enabled student information system might look like.
4. an examination of current XML standards in areas that affect student information systems.

In conclusion we will look at the various business models that are supported by XML enabled architectures.

1 What is XML?

SGML (Standard Generalized Markup Language) is the ISO standard for markup languages. The two most important markup languages that conform to this standard are XML (eXtensible Markup Language) and HTML (Hypertext Markup Language). There are four aspects of the XML language that deserve a brief mention before we attend to the different ways XML may impact the development of student information systems:

- The basic syntax of an XML document.
- Document Type Declarations or, DTD’s.
- XML schemas.
- XML stylesheets, or XSL.
There are two major differences between HTML and XML.

- HTML tags describe the format of a document, whereas in XML the tags describe the data itself.
- HTML tags are predefined, whereas XML tags are user defined.

**Figure 1**
A simple HTML document

```html
<HTML>
<BODY>
<B>Chemistry 230</B>
</BODY>
</HTML>
```

This would cause a web browser to display the phrase **Chemistry 230** in a bold typeface. The same data could be described in an XML (see figure 2).

**Figure 2**
A simple XML document

```xml
<COURSE>
  <SUBJECT>Chemistry</SUBJECT>
  <NUMBER>230</NUMBER>
</COURSE>
```

In an XML document you can define which tags are allowed, the order in which they may appear and, the kinds of data they may contain. These rules can be expressed in the document’s Document Type Declaration or, DTD.

**Figure 3**
Our simple XML document is now prefaced by a DTD

```xml
<?XML VERSION = "1.0" ?>
<!DOCTYPE DOCUMENT [ 
  <!ELEMENT DOCUMENT (COURSE)*> 
  <!ELEMENT NAME (SUBJECT,NUMBER)> 
  <!ELEMENT SUBJECT (#PCDATA)> 
  <!ELEMENT LASTTNAME (#PCDATA)> 
]>

  <COURSE>
    <SUBJECT>Chemistry</SUBJECT>
    <NUMBER>230</NUMBER>
  </COURSE>
```

The DTD is enclosed in square brackets [ ]. The DTD illustrated in figure 3 contains an element called COURSE which may be repeated multiple times (this is indicated by the asterisk *). A COURSE contains two other elements, SUBJECT and NUMBER and these two elements may contain character data (PCDATA). An XML document is considered valid if it conforms to the rules specified in the DTD.

There is another, and much more powerful way of describing an XML document, namely an XML schema. A schema is itself an XML document. Consequently, new tags can be invented to describe new data constraints as they arise. A DTD has a closed, or fixed, syntax. An XML schema is, by contrast, infinitely extensible. Because of this there is a strong presumption in favor of using schemas rather than DTD’s to describe complex data structures such as EDI transactions.
An XML schema that describes our simple XML document

```xml
<schema>
  <element name="Course" minOccurs="0" maxOccurs="*">
    <type>
      <element name="SUBJECT" type="string"/>
      <element name="NUMBER" type="string"/>
    </type>
  </element>
</schema>
```

The last aspect of XML that needs a brief mention is the XML stylesheet, or XSL. Although XML is primarily a language for describing data and HTML is a language for describing display formats, it is easy to see how one could create a bridge between the two by specifying the display formats that apply to elements within an XML document. This is what XSL does. The function of XSL is nicely expressed in the Microsoft developer's network: “The XSL transformation language enables display of XML by transforming XML into grammar and structure suitable for display—for instance, into HTML.”

A simple XSL stylesheet

```xml
<?xml version='1.0'?>
<xsl:template match="/">
  <HTML>
    <BODY>
      <B>
        <xsl:value-of select="SUBJECT"/>
        <xsl:value-of select="NUMBER"/>
      </B>
    </BODY>
  </HTML>
</xsl:template>
```

If we applied the stylesheet in figure 5 to our simple XML document it would cause the word Chemistry to be substituted in the place of SUBJECT and 230 to be substituted for NUMBER. In other words our simple XML document would be transformed into the simple HTML document in figure 1.

XML is merely one among many ways of describing data. The most formal and academic is UML (Universal Modeling Language). Objects that are stored on a relational database are described with DDL (Data Definition Language). The same object can be described in object oriented languages like Java\(^\text{\textsuperscript{10}}\) and C++. EDI (Electronic Data Interchange) is another way of describing complex data objects. So why bother about yet another standard for describing data? The answer is probably that XML exhibits the same simplicity, openness and robustness as HTML:

- It is very easy to understand. The tags are all in English text (or relatively obvious acronyms).
- The tags are inside the document itself. This means you do not have to refer to a separate document or object in order to understand the document.
• It is an open standard. Software developers can easily write to the standard. It is very much a case of building a road and the traffic follows.

2 XML technologies

XML support is now built into a number of standard commercial technologies. Programs that can read and manipulate XML documents are called parsing engines and many are freely available on the Internet. XML is now used extensively in publishing software as it can be used both to describe data components and complex formatting directives. Relational database products can read and save XML documents. All these products are currently available to anyone who is engaged in the custom development of information systems.

Parsing engines.
The most basic XML technology is the parsing engine. Parsing engines are typically written in Java. All major software companies have built XML parsing engines (Sun, IBM, Oracle and Microsoft) and many of them are freely available on the web. A DOM (Document Object Model) parsing engine will take an XML document and read it into a object so that it can be manipulated. If we had saved our trivial XML document (see Figure 2) and saved it to c:\xml\course.xml, then the following line in a Java program would create a document object that could then be manipulated.

```
Document document = SimpleDOMParse("c:\xml\course.xml")
```

If, for instance, we wished to change the course description from chemistry to biology we could say
```
findReplace(document, 'SUBJECT', 'Chemistry', 'Biology')
```

And then we could save the new document that would now look like this:

```
<DOCUMENT>
  <COURSE>
    <SUBJECT>Biology</SUBJECT>
    <NUMBER>230</NUMBER>
  </COURSE>
</DOCUMENT>
```

Database extenders
Both IBM and Oracle have added XML support to their database products. It is easy to see how a document object model can be mapped onto a relational data model. The trivial XML document that we have been using as our example could be mapped to a database table called COURSE that contained two columns SUBJECT and NUMBER. A database extender could store the document directly in the database table, and conversely, extract the document from the table.

Publishing Software
Publishing software products like AdobeFramemaker support XML. This means that if you are publishing your University Calendar with one of these products you could define a course component like the one described in Figure 2. All references to that component would match the XML description.
Web browsers

Xml support is now built into web browsers. If you opened our simple XML document in IE5.0 you would get an interactive tree that you can manipulate (see Figure 7). IE5.0 also supports XSL so you can use XML with stylesheets to display documents.

Figure 7
What you would see if you opened our simple XML document in IE 5.0. You can collapse the tree by clicking on the minus sign.

The interesting thing is that all these different technologies can all understand and work with the same XML document. The publishing software that is used to publish the University Calendar can understand and render the same tags that are used by a web browser in conjunction with an XSL stylesheet to display course information on the web. An EDI transmission of a transcript to another institution would work with the same tags. Lastly, the same course information could be saved to a database (and retrieved) in exactly the same format using database extenders. Figure 8 is a conceptual model of these interrelationships.

Figure 8
Different XML aware software products can all work with the same XML document.
3

XML and Student Systems

Perhaps the most obvious place to envisage the use of XML in student systems would be to replace EDI transactions. The most well known EDI transaction is the T130, or, student transcript (see Appendix A). The EDI specification describes the data elements for a student transcript and the constraints that apply to these elements. This includes information such as:

- Data types (e.g. text, numeric, date)
- Number of times a field can be repeated.
- Whether a field is mandatory or optional.
- The domain of possible values (e.g. Male or Female)
- The relationship of this element to other elements (e.g. one-to-many, many-to-one, one-to-one)

The syntax and semantics of student transcripts have already been established. It is just a question of how these should be represented. The EDI representation is actually in English and is not machine readable (see Appendix A). The student transcript could equally well be described by a UML diagram. An XML schema is just another way of capturing the syntax and semantics of a student transcript. The meaning of all the fields and segments have all been agreed upon. The advantage of an XML representation is that there are already numerous technologies (outlined in Part 2) that would be able to read it and work with it.

Figure 9
An XML schema fragment that describes elements 1251, 1068 and 1067 of the EDI transaction T130 (see Appendix A)

```xml
<schema>

..................
<type name="Names">
<element name="demographicInformation " minOccurs="1" maxOccurs="1">
<element name="birthdate" type="string"/>
<element name="gender" type="string"/>
<element name="maritalstatus" type="string"/>
</element>
</type>
</schema>
```

Obviously it would be more complex than this. We need to be able to indicate whether something is optional/mandatory, specify the domain of allowable values (M=male, F=female) and other constraints. But as a schema is itself an XML document, we should be able to invent tags for any additional constraints that we need to consider. The actual transcript document that could be validated against the grammar defined in the schema would contain the tags specified in the schema (see Figure 10).
If we take the main EDI transactions, the student transcript (T130), the admission application (T189) and the test results (T138), they contain all the basic components for a comprehensive data model of a student information system. Student demographics, courses and tests are the basic “stuff” of the SIS. At UBC we have been redeveloping our SIS entirely on the internet and the kinds of things that we pass across the network are semantically identical to the components of an EDI transaction. Our new internet based SIS is really a microcosm of the EDI world. We support a number of different clients:

- Other servers.
- Applets
- Applications
- HTML browsers

The same basic piece of information can be delivered to all four clients. A student can download a timetable into palm pilot. An administrative SIS user can see the timetable in the SIS application. A student can view it through a browser as an applet or as an html page. In a similar vein, another server (hosting a different application) can get a student’s timetable by making a simple http request. In each case the same basic piece of information is being passed across the network (and, conceptually speaking, in each case it is a subset of an EDI transaction).

Unfortunately our applications are using different wire level protocols (RMI, https) and within each protocol different data formats are being used (see Figure 10):

- When the server talks to our SIS application the student timetable is delivered as a set of Java objects using RMI
- When the timetable is delivered to a browser it is passed as html text over http
- When the timetable is passed to another server that hosts course delivery software (WebCT) it is passed as a tab delimited string over http
- When the timetable is passed to an applet is passed as java objects over the http object stream.
The tower of Babel
Different wire level protocols (RMI, https) are used. The information is passed in a variety of formats: tab-delimited text, html, and serialized Java objects. Yet in each case the content is the same: students and their grades.

The ideal world
Everyone uses the same wire level protocol (https) and all objects are described in the same language: XML

Ideally, the same wire level protocol should be used in each case and the data should always be presented in the same format. Further, the syntax and semantics of the objects that are passed between servers and clients within the SIS should be identical to those of the transactions that are passed between institutions. An EDI student transcript that was received as an XML document contains exactly the same components as a grades upload from the Faculty of Arts to the main SIS server. Metaphorically speaking, EDI between institutions could be described as the international trade in student objects whereas the SIS is the local trade. But both international trade and local trade should use exactly the same containers and the same transportation methods. In practice, using the same containers would mean referring to a URL that hosted a schema that the various interested parties could subscribe to.
4

Current XML standards

In an ideal future, there would be industry-wide standards for describing things like student transcripts in XML. There would be an XML schema for the EDI T130 student record transaction. This in turn would be built out of standard XML building-blocks such as institution identifiers, course descriptions and student demographics. When a faculty uploaded grades to the central SIS, the transactions would be built out of exactly the same XML building blocks as a student transcript transmitted from one institution to another. In short, all clients and servers would speak the same language. If anything like this is ever going to happen then everyone needs to follow the same set of standards.

XML standards are developed by W3C. The current standard for XML 1.0 is published at their site www.w3.org. A working draft for XML schemas is available and the committee working on the draft intends to submit the specification for publication in March 2000.

EDI standards are developed and maintained by X12 which is part of the American National Standards Institute¹². Transaction sets for higher education are maintained in conjunction with AACRAO (the American Association of Collegiate Registrars and Admissions Officers). Work is being done on XML equivalents for EDI transactions.

Of course standards are volatile and the industry is moving very fast. Six months from now the picture may be quite different. Many projects cannot wait for the official publication of standards. Nonetheless, there is an obvious advantage if developers work with existing drafts of standards (rather than reinventing the wheel each time). Even if there is not complete agreement over the exact meaning of tags (and, realistically speaking, there probably never will be) as long as everyone is using XML in more or less the same way, everyone will be using different dialects of the same language instead of using entirely different languages.
The Internet opens up possibilities of conducting business in previously unimagined ways. At UBC this has caused us to rethink how we deliver student services. Many of the constraints of time and place have disappeared. The slogan “Anytime, Any place” has already materialized in a number of ways. This is just the tip of the iceberg. We want to dramatically extend the kinds of business we conduct over the Internet (fees, financial aid, library loans etc etc). And as we increase the number of different kinds of things students can do over the Internet so we need to give our students a personalized web presence (or, portal) that allows them to organize all of this in a meaningful way.

From business perspective this is really about collaborative efforts and partnerships. Different service providers on campus (such as student-services, the library and athletic services) and, obviously, instructors, need to be able to exchange information in a meaningful way. An instructor who is using Internet classroom management tools needs need to subscribe to a service published by the Student Information System that gives current class enrollments. A student who wants to see a consolidated financial statement needs to subscribe to statements published by the library (library fines), athletic services, parking and security as well as the usual fees and financial aid. The underlying architecture for this is a group of servers that communicate in the same language. A simple and robust solution to the problem is for all servers to accept requests expressed in XML and to deliver services as XML documents over https (as illustrated in Figure 10).
This is what the element summary of the T130 student transcript EDI transaction looks like. As you can see, description of the syntax and semantics of an EDI transaction is actually stored in a separate English-like document. This document is the specification for a specialized parsing engine that can read an EDI transaction. In XML the syntax and semantics are captured within the document and there are numerous commercial applications that can parse the document.

<table>
<thead>
<tr>
<th>Ref</th>
<th>Id</th>
<th>Element Name</th>
<th>Req</th>
<th>Type</th>
<th>Min/Max</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMG01</td>
<td>1250</td>
<td>Date Time Period Format Qualifier (Format for Date of Birth)</td>
<td>X</td>
<td>ID</td>
<td>2/3</td>
<td>Used</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Description: Code indicating the date format, time format, or date and time format</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note to User: The date of birth is a useful identifier on the transcript for those institutions which do not use the Social Security number as an identifier.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Code</td>
<td>NAME</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CM</td>
<td>Date in Format CCYYMM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>CY</td>
<td>Year Expressed in Format CCYY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>D8</td>
<td>Date Expressed in Format CCYYMMDD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB</td>
<td>Date Expressed in Format MDDCCYY</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>MD</td>
<td>Month of Year and Day of Month Expressed in Format MDD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| DMG02| 1251 | Date Time Period (Date of Birth) | X   | AN   | 1/35    | Used  |
|      |      | Description: Expression of a date, a time, or range of dates, times or dates and times |
|      |      | Note to User: The date of birth is indicated here in the format identified in DMG01. |

| DMG03| 1068 | Gender Code | O   | ID   | 1/1     | Used  |
|      |      | Description: Code indicating the sex of the individual |
|      |      | Code | NAME                                      |      |      |         |       |
|      |      | F    | Female                                    |      |      |         |       |
|      |      | M    | Male                                      |      |      |         |       |
|      |      | U    | Unknown                                   |      |      |         |       |

| DMG04| 1067 | Marital Status Code | O   | ID   | 1/1     | Used  |
|      |      | Description: Code defining the marital status of a person |
|      |      | Code | NAME                                      |      |      |         |       |
|      |      | A    | Common Law                                |      |      |         |       |
|      |      | B    | Registered Domestic Partner               |      |      |         |       |
|      |      | D    | Divorced                                  |      |      |         |       |
|      |      | I    | Single                                    |      |      |         |       |
|      |      | K    | Unknown                                   |      |      |         |       |
|      |      | M    | Married                                   |      |      |         |       |
|      |      | R    | Unreported                                |      |      |         |       |
|      |      | S    | Separated                                 |      |      |         |       |
|      |      | U    | Unmarried (Single or Divorced or Widowed) |      |      |         |       |
|      |      | W    | Widowed                                   |      |      |         |       |
|      |      | X    | Legally Separated                         |      |      |         |       |

| DMG05| 1109 | Race or Ethnicity Code | O   | ID   | 1/1     | Used  |
|      |      | Description: Code indicating the racial or ethnic background of a person; it is normally self-reported; Under certain circumstances this information is collected for United States Government statistical purposes |
|      |      | Code | NAME                                      |      |      |         |       |
|      |      | 7    | Not Provided                              |      |      |         |       |
|      |      | A    | Asian or Pacific Islander                 |      |      |         |       |
|      |      | B    | Black                                     |      |      |         |       |
|      |      | C    | Caucasian                                  |      |      |         |       |
|      |      | Etc  | etc                                        |      |      |         |       |
1 See http://www.oracle.com/xml/.

2 Actually, IBM’s product is an add-on to the DB2 Universal Database. It is called the DB2 XML Extender. See http://www-4.ibm.com/software/data/db2/extenders/xmlext/.

3 IE5.0 can display an XML document as a tree structure so you can traverse the data. It can also display an XML document according to the formatting prescribed in an XSL stylesheet. See http://msdn.microsoft.com/xml/XSLGuide/

4 Microsoft heads a forum for businesses interested in using XML for EDI. It is called Biztalk and is available at http://www.biztalk.org. There are also non-profit organizations developing XML repositories for EDI. A good example is OASIS (see http://www.oasis-open.org) which is backed by Sun, Adobe, IBM, Boeing and others. EDI/XML conversion efforts are not limited to North America as there is a European XML/EDI Pilot Project (see http://www.cenorm.be/isss/workshop/ec/xmledi/isss-xml.html).

5 As of 31 December 1999, the World Wide Web Consortium (W3C) had 379 members ranging from private corporations like IBM to international publicly funded organizations like CERN. Full membership costs US$50,000 annually. At the end of 1999 the W3C team consisted of 58 people. The W3C website (www.w3c.org) is hosted by the Massachusetts Institute of Technology Laboratory for Computer Science [MIT/LCS] in the United States, at the Institut National de Recherche en Informatique et en Automatique [INRIA] in France, and at the Keio University Shonan Fujisawa Campus in Japan.

6 We have a Java program that reads our curriculum database and produces an XML extract that looks like this:
   ```xml
   <?xml version="1.0" encoding="UTF-8" ?>
   <subject>
       <head>Accounting</head>
       <code>BAAC</code>
   </subject>
   <faculty>Faculty of Commerce and Business Administration</faculty>
   <course><number>BAAC 500</number>
       <credits>(1.5)</credits>
       <crstitle>Financial Reporting</crstitle>
   </course>
   ```
   The extract is then imported into Adobe FrameMaker+SGML which is our main publications software.

7 There are a number of good descriptions of SGML at www.w3c.org. See, for instance, http://www.w3.org/TR/html401/intro/sgmltut.html

8 There are countless books on XML. Books in Print currently lists 80 books on XML. For this paper I used XML complete by Steven Holzner (McGraw-Hill 1998).

9 See http://msdn.microsoft.com/xml/XSLGuide/

10 For a theoretical discussion of mapping XML schemas to Java objects see: An XML Data-Binding Facility for the Java Platform. For a broader discussion of the synergy between Java and XML see: Portable Data / Portable Code: XML & Java Technologies. Both papers are at http://java.sun.com/xml/white-papers.html

11 Adobe is probably best known desk-top publishing company. The specification for FrameMaker is at http://www.adobe.com/products/framemaker/prodinfosgml.html
A major European supplier of publishing software is Balise. Their product, DualPrism is used by over 200 companies including Ericsson, Nokia, Airbus Industrie and Texas Instruments. XML lies at the heart of their publishing technology (see http://www.us.balise.com)
There are a number of different organizations involved in the maintenance of EDI standards. The most important is X12 (see http://www.x12.org) that was chartered by the American National Standards Institute (ANSI) to develop standards for the electronic exchange of business transactions. The transaction sets that are of interest to higher education have been developed and maintained in conjunction with AACRAO (the American Association of Collegiate Registrars and Admissions Officers, see http://www.aacrao.org). The transaction sets include:

- Student Transcript (Transaction Set 130)
- Acknowledgment of a Student Transcript (Transaction Set 131)
- Request for a Student Transcript (Transaction Set 146)
- Response to a Request for a Student Transcript (Transaction Set 147)
- Course Inventory (Transaction Set 188)
- Application for Admission (Transaction Set 189)
- Educational Testing Results Report and Request (Transaction Set 138)