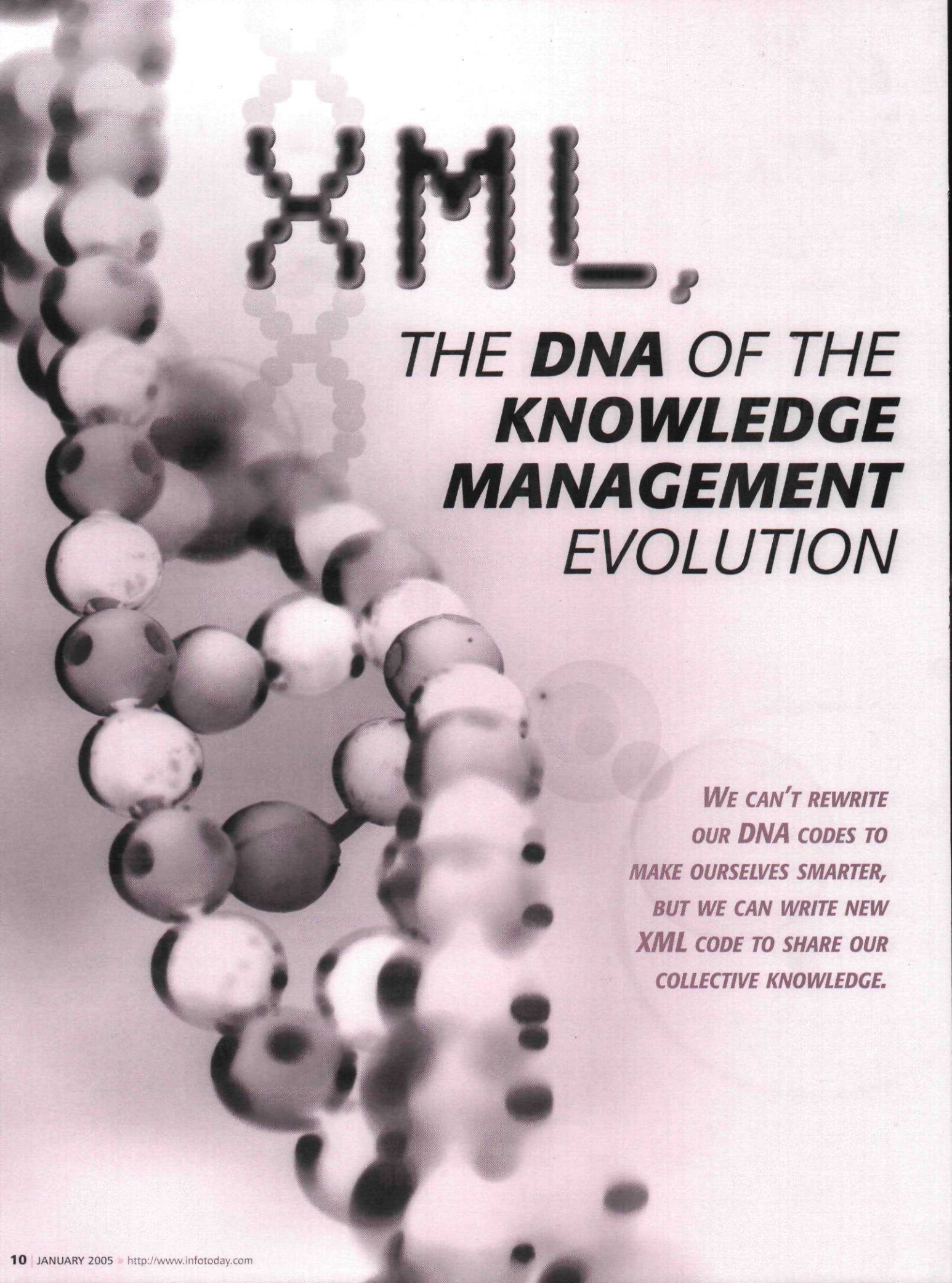

XML, THE DNA OF THE KNOWLEDGE MANAGEMENT EVOLUTION

WE CAN'T REWRITE OUR DNA CODES TO MAKE OURSELVES SMARTER, BUT WE CAN WRITE NEW XML CODE TO SHARE OUR COLLECTIVE KNOWLEDGE.

BY JAN HAYES, CHRISTINA STOLL, AND DAWNE TORTORELLA

The North Suburban Library System (NSLS) is one of nine multitype library systems that is funded through a grant from the Illinois State Library. Our library systems provide services and support for Illinois libraries—public, school, academic, and special. The specifics on how we implement technology to deliver those services and support varies greatly with each system. The members represent a great variety in our species as far as the way we adapt technology.

In 2002, the Illinois State Library funded $20,000 for a cooperative Knowledge Management Project among NSLS, Lincoln Trails Library System (LTLS), and Northern Illinois Library System (now a part of the Prairie Area Library System, or PALS). The purpose of our grant was to emphasize the common work that the systems do to serve all the libraries by creating a prototype of a knowledgebase to share common documents and the collective wisdom of the staffs in the system. Working with technology consultants BellCow, Inc., the team developed a statewide knowledgebase about continuing education events as the initial content.

Jan Hayes, NSLS's assistant director, was the project manager, and Christina Stoll, knowledge management specialist, led the project team in defining and codifying the knowledgebase. Dawne Tortorella and David Nelson from BellCow developed the knowledge management application and provided training.

Instinctively, we knew that XML would be the key building block of our knowledge management "life-form." And it is certainly evident that XML skills in libraries will be a factor in the survival of the fittest. We just had to unravel the DNA-like code to understand how to embrace our similarities and appreciate our differences without sacrificing our individual ways of doing business.

No Magic Code Structure: The Challenge and Solution

Early in the project it became apparent that we had reached the chasm between reality and vision in our implementation strategy to create a searchable repository of XML documents. XML, by itself, doesn't do anything, although it is undoubtedly the most promising standard for information storage and interchange. As Kyle Banerjee warned in his article "How Does XML Help Libraries?" (*CIL*, Sept. 2002, http://www.infotoday.com/cilmag/sep02/Banerjee.htm), "It's important to recognize that XML only provides a structure for storing information. It does not say anything about how information is displayed, it does not create links that people can click on, and it doesn't bring information resources together by itself." Storing data in XML format is easy; being able to update, search, and display that information proved to be the technical challenge.

The Project's Multiple Goals

In addition to the project goal of sharing data in standard format among several of the library systems in Illinois, we wanted to work collaboratively on a project that would help develop understanding and skills in XML. But the first step was to decide what topic we'd use to populate our initial knowledgebase. We considered several content-heavy resources, including job descriptions, policies, procedures, and consulting expertise as possibilities for our starting knowledgebase collection. These are ideal choices and are earmarked for future projects; however, our tight time frame required us to choose information that could be quickly codified and pulled into a repository.

Since every regional system tracked continuing education (CE) events and had an interest in sharing this information throughout the state, we selected CE events as a starting point in knowledge management and XML collection. We weren't just looking to build a combined calendar-of-events application; the real benefit of sharing this information is to track program trends, share instructors, and collaborate on professional development programming.

The project team reviewed assumptions and project requirements. These were the goals:

1. To create a method for sharing common information that can be searched, updated, and displayed

2. To allow library systems to retain their current processes, data structures, and database technologies for storing events and processing registrations

3. To develop a shared element structure

4. To avoid creating a static Web site and Web pages and, instead, have the shared data serve dynamic content

5. To provide access for search and retrieval, importing, adding, and updating information via a Web browser interface

6. To use XML as the underlying technology for data definition, storage, and display

7. To make the XML technologies that were used apparent so that the application could serve as a case study to demystify the critical components of an XML solution

»

Cracking the Code: The Technology We Used

At the time the project was launched, the project team and consultants conducted a thorough analysis of available XML tools and services. The project team specifically looked at open source solutions to handle the native XML database (repository) needs, indexing, and importing/updating of the native XML documents. A few open source projects were underway, but they had not developed to the point required to meet our implementation deadline in fall 2002. Commercial products did, however, offer solutions that could be customized to satisfy the specific project goals within the time constraints we faced. IXIASOFT's TEXTML Server is the cornerstone product used in the project. It provides a repository that stores XML files in their native format. TEXTML also provides element-level indexing, supporting full-text queries against the XML content housed in the repository.

The consulting team developed a customized application to provide bulk file import, file purging based on date range, and search/display of data from the XML repository. Since we'd have contributions from each library system in the state, it was important to have document-level authentication. We wanted to build the system so that all XML documents could be viewed by anyone, but only the document owner (library system in our case) would be allowed to delete or update an XML file.

Altova's Authentic Browser Plug-In interfaces with the TEXTML Server, allowing XML documents to be retrieved from the repository and edited online via a browser plug-in. While the majority of XML files are imported in bulk via the application, the Authentic Browser Plug-In provides the added benefit of enabling new events to be input directly, if desired. We employed Altova's XMLSpy to model the XML schema and style sheet (XSLT) used to display search query results. The consultant invited members from all the library systems to a day of hands-on training in using XMLSpy. All participants got a chance to experiment with the XML application and to see that all pieces are based on a common structural component, just like the nucleotide bases of DNA.

Our goal of users gaining knowledge in the use and application of XML is further facilitated by the TEXTML Server's "under the hood" feature. This feature allows people to peruse the XML used to query, sort, and format the information. The solution uses XML throughout all processes and serves as a valuable training and demonstration tool in the applied use of XML data.

Genetic Re-Engineering: Implementation Steps

Our technical list of tasks was to develop a data-type definition (DTD) to describe the essential data elements in a continuing education event. In keeping with the project goal of maximizing the use of XML technologies, this definition was developed as an XML schema. The schema modeling was completed using Altova's XMLSpy, which allowed the committee to visualize the structure and to quickly develop the schema. In an afternoon session, literally three members of the project team were able to drag, drop, and define our data model visually. Behind the scenes, the more complex XML schema was developed by the software. Using XMLSpy we were able to concentrate on the information we wanted to map instead of worrying about the syntactical rules of the XML schema.

The XML schema became the basis for generating XML files from existing database content and for building the searchable indexes in the XML repository. In reality, the TEXTML Server and the customized application do not enforce adherence to this schema (except in the case of adding new events through the plug-in described below). At first this seemed like anarchy that would lead to genetic XML mutations. But, in practice, it enabled the library systems to retain elements deemed of importance in their individual organizations without endorsement by the entire group.

The Altova XMLSpy interface is illustrated in Figure 1 and interested readers can see the resulting XML schema at http://www.nsls.info/km/CEEvents.xsd.

Mapping the Past

The library systems each had internal procedures and technologies in place for maintaining their own contin-

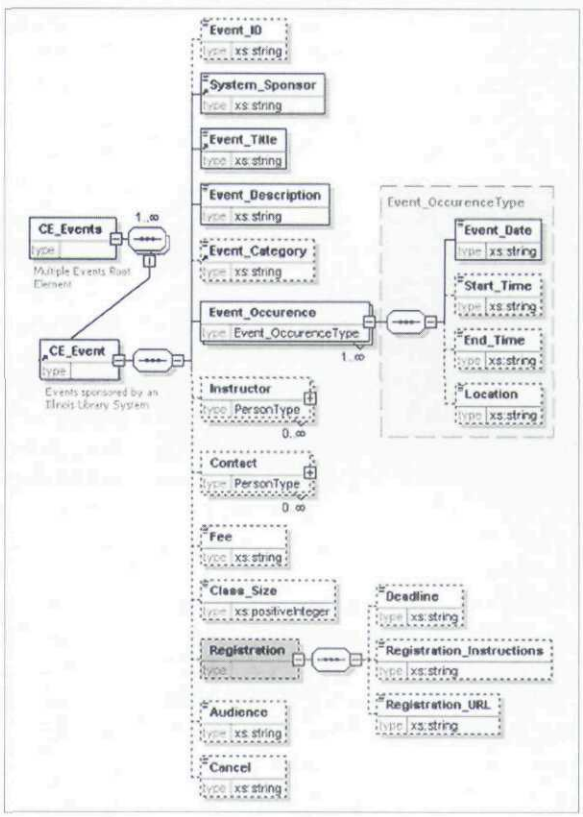

Figure 1: Here's a view of the Altova XML schema.

uing education workshops and registration for these events. To extract the common elements of events, the NSLS application developer and consultants wrote customized programs, each tailored to specific library systems, to read their existing databases (SQL Server, Microsoft Access, Oracle, FileMaker Pro) via ODBC and ASP/PHP programs. The purpose of the customized programs was to correlate the existing data structure of each particular library system with the elements defined for our project, query the database based on a date range, and output an XML file of CE events that matched the project schema.

Adhering to proper XML rules of coding posed the most difficult challenge in the data extraction process. From years of inputting data directly into relational databases, "work-arounds" had been developed, including cut-and-paste operations from word processing programs that use special characters and direct entry of HTML tags into the data content. Actually, both can be supported in XML, if the XML syntax rules are followed. The conversion programs caught most of the special character problems and changed the character to the valid entity character representation (e.g., "&" is converted to "&"). Embedded HTML can be inserted without incident if the tags are properly nested, if empty tags are correctly represented, and all tags are closed (e.g., bold italicized text;
 instead of
; with a closing). If the HTML used is XHTML-compliant, it is imported with no problem. (Our recommendation is to always use well-formed XHTML if you're embedding HTML tags in any database or Web page.)

The code segment shown in Figure 2 illustrates the process of reading from a database and writing an XML file in ASP code.

The file resulting from running the extract program is an XML document that validates against the schema. The XML seen in Figure 3 shows an example of the output generated.

In addition to facilitating a bulk load import of events formatted from existing database records, we needed to enable on-the-fly entry and editing of events.

```
<%
  Set oRS = oConn.Execute(oSQL)
  Response.ContentType = "text/xml"
%>

<?xml version="1.0" encoding="UTF-8"?>
<System_Events xmlns:xsi="http://www.w3.org/2001/XMLschema-instance"
xsi:noNamespaceschemaLocation="CEEvents.xsd">

<% Do While Not oRS.EOF
' validate data from database and set string values for XML
' [validation code goes here]
' [format strInstructor, strContact, strFee, strLimit, strRegURL]
' Format XML Output
' Loop through repeating event dates, each unique date for an event is a repeating XML element
%>
  <CE_Event ID="<%= iCourse  %>">
  <System_Sponsor>CLS</System_Sponsor>
  <Event_Title><% = ReplaceBadChars(oRS("Title")) %></Event_Title>
  <Event_Description><% = strDesc %></Event_Description>
  <Event_Category><%= strCategory %></Event_Category>
<%
  iSavedCourse = oRS("CourseID")
  iCurCourse = oRS("CourseID")
  Do While (iCurCourse = iSavedCourse) And (Not oRS.EOF)
%>
    <Event_Occurence>
      <Event_Date><%=(oRS("StartDate"))%></Event_Date>
      <Start_Time><%=(oRS("StartTime"))%></Start_Time>
      <End_Time><%=(oRS("EndTime"))%></End_Time>
      <Location><%=(oRS("Location")%></Location>
    </Event_Occurence>
<%
    oRS.moveNext
    If Not oRS.EOF Then
      iCurCourse = oRS("CourseID")
    End If
  Loop
%>
  <% = strInstructor %>
  <% = strContact %>
  <% = strFee %>
  <% = strLimit %>
  <Registration>
    <Registration_Instructions>Register online at: </Registration_Instructions>
    <Registration_URL><%= strRegURL %></Registration_URL>
  </Registration>
  </CE_Event>
<% Loop %>
</System_Events>
```

Figure 2: This is a section of an ASP program that we created to generate XML from an existing database.

```
<?xml version=î1.0î encoding=îUTF-8î?>
<System_Events xmlsns:xsi=http://www.w3.org/2001/XMLschema-instance
xsi:noNamespaceschemaLocation=îCEEvents.xsdî>
<CE_Event ID=î490î>
<System_Sponsor>NSLS</System_Sponsor>
<Event_Title>The Union List: Searching and Adding Serial Holdings</Event_Title>
<Event_Description>This workshop provides training in the OCLC Union List Service including
a discussion of union list features and benefits, offline products, and SILO requirements
for participation. Participants will learn how to create a Union List session with OCLC,
how to search for and identify serial records, and how to create, edit, and delete a local
data record. The option to batchload local data records will also be discussed. This workshop
provides hands-on experience. Prerequisite: #050 (Searching WorldCat) or equivalent OCLC
searching experience.
</Event_Description>
<Event_Occurence>
  <Event_Date>10/28/2004</Event_Date>
  <Start_Time>9:30:00 AM</Start_Time>
  <End_Time>4:00:00 PM</End_Time>
  <Location>NSLS - Computer Lab, 200 West Dundee, Wheeling, IL 60090</Location>
</Event_Occurence>
<Contact>
  <FirstName>Ruth</FirstName>
  <LastName>Downey</LastName>
  <Phone>847-459-1300, ext. 7121</Phone>
  <Email>rdowney@nsls.info</Email>
</Contact>
<Fee>20</Fee>
<Class_Size>12</Class_Size>
<Registration>
  <Registration_Instructions>Register online.</Registration_Instructions>
<Registration_URL>http://www.nsls.info/ce/ClassDetail.asp?course=490</Registration_URL>
</Registration>
</CE_Event>
<CE_Event ID="95">
   [data elements inserted here for second event, etc]
</CE_Event>
[more events...]
</System_Events>
```

Figure 3: This is a sample of the XML that we generated with ASP. Now it's useful data for a bulk load.

The IXIASOFT TEXTML Server's XML data repository interfaces with Altova's Authentic Browser Plug-In (see Figure 4). Using this plug-in, XML documents stored in the TEXTML Server repository can be retrieved and edited using a browser plug-in. Colleagues can enter new events using the same approach.

The TEXTML Server stores the native XML files in a repository. The repository is indexed based on custom indexes defined using the XML schema elements. The index definition is based on element XPath specification and various data types including dates, along with string and word characteristics. Once defined, the indexes are dynamically updated whenever a document in the repository is modified, added, or deleted. The structure of the XML is read and indexed based on the index definition (see Figure 5). This allows searching across the native XML files housed in the repository.

XSLT, eXtensible Stylesheet Language Transformation, plays an important role in the display of query results. When a query is made against the repository, multiple documents (in our case, multiple continuing education events) are returned. The application reads these returned XML elements and formats them for display by transforming the XML into HTML, which can be viewed on any standard Web browser. The actual data delivered to the browser window is the transformed HTML. The formatting and display attributes shown in the browser are derived from the XSLT rules.

Figure 6 illustrates how an event is displayed after it is transformed via the XSLT.

You can see the complete XSLT used to transform the XML data at http://www.nsls.info/km/CEEvents.xsl.

XML as Genetic Code for Knowledge Management

In our project, XML proved itself both extensible and adaptable. While the body of knowledge selected for the project consisted of continuing education events sponsored by the Illinois library systems, the final technical solution works equally well with any body of knowledge.

Soon after the project was launched, colleagues made requests to extend the sorting capabilities of the result set, to make the location of events a searchable element, and to change the formatting of the transformed HTML to display the complete library system name of the sponsoring agency. All modifications were easily accommodated by additional index definitions in the TEXTML Server and modifications to the XSLT.

Some systems quickly tried to push the envelope by including additional elements in their continuing education XML events. This had no adverse affect on the shared repository and allowed individualization of the XML content, as long as the essential common elements defined in the schema and used for query retrieval were included in the XML.

We are now considering additional bodies of knowledge to include in the

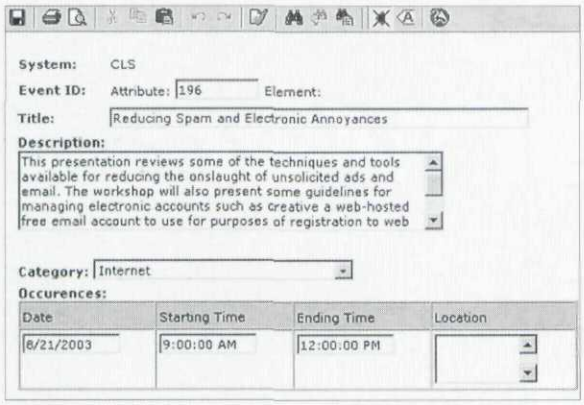

Figure 4: Altova Authentic Browser Plug-In for editing and adding XML

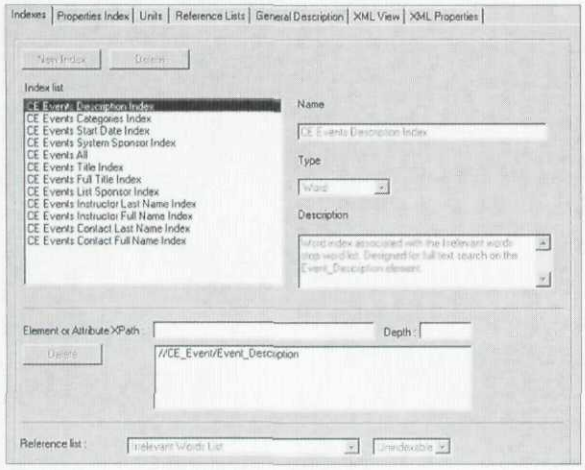

Figure 5: Indexes defined in TEXTML Server

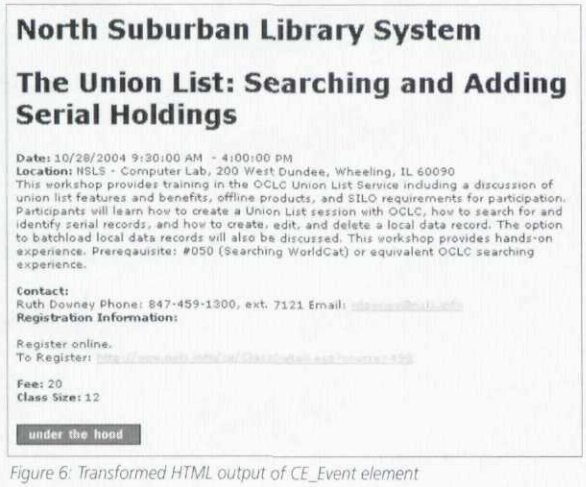

Figure 6: Transformed HTML output of CE_Event element

repository. The core technologies and applications can easily be extended to accommodate any new collection of information. Given the indexing capabilities of the TEXTML Server, we can define information schemas that enable vastly different bodies of knowledge to coexist within one knowledgebase.

Our experiences reaffirm the benefits of XML and its extensibility.

It Takes More Than Science: You Need Understanding

The technology alone was not sufficient to make the project a success. It continues to be important for us to understand the essence of XML and how applications are built around this robust standard. As we saw, creating XML data was pretty easy. Creating an application based on XML standards throughout was a challenge we are happy we tackled. Through the training that was included in the project, we were able to explain and demonstrate core XML genetic code, including XML schema for defining the data structure, transformation of relational database information into raw XML data, XSLT-eXtensible Stylesheet and Transformations for converting the raw XML data into a browser-friendly HTML output, and XPath for defining the element relationship and indexes to support query. One of the most successful outcomes of the project was that systems' staffs have gone on to use XML in other technology projects.

Our experience and growing comfort in using XML in library applications makes us confident that our library systems can crack technology's genetic code and stay a step ahead of the evolutionary curve.

Company Contacts

Altova, Inc.
900 Cummings Center
Suite 314 T
Beverly, MA 01915
(978) 816-1600
http://www.altova.com

IXIASOFT
75, Queen
Suite 4600
Montreal, Quebec
H3C 2N6
CANADA
(514) 279-4942
http://www.ixiasoft.com

Jan Hayes is assistant director at the North Suburban Library System in Wheeling, Ill. She holds an M.L.I.S. from the University of Illinois, Urbana. She has more than 18 years of library experience. For the past decade, she's focused on managing and implementing technology-based solutions for NSLS. Her e-mail address is jhayes@nsls.info. Christina Stoll is the knowledge management specialist for the North Suburban Library System in Wheeling, Ill., and spearheads its Knowledge Management Initiative. She holds an M.L.I.S. from the Graduate School of Library and Information Science at Dominican University in River Forest, Ill. She has more than 10 years of experience working in a variety of libraries, focusing on knowledge management and technology. Her e-mail address is cstoll@nsls.info. Dawne Tortorella is president of BellCow, Inc. in Riverside, Ill. She holds an M.S. in computer science from DePaul University in Chicago. She has more than 25 years of experience in systems design and architecture and has worked with libraries of all types for more than 15 years. Her e-mail address is Dawne.Tortorella@BellCow.com.