Migrating from PDF to XML based ETDs:
a report on a work in progress at the University of Maine

Marilyn Lutz
Director, Information Technology Planning
Fogler Library, University of Maine
lutz@maine.edu

Abstract
What advantage is there for your institution to move from PDF (Adobe Acrobat's Portable Document Format) to XML, the Extensible Markup Language as the standard format for ETDs? There may not be an immediate benefit from adopting XML. Over time there may be the distinct advantage of making your scholarly collections more visible and more accessible to a broader community of students and scholars.

This paper reports on work in progress at the University of Maine Library to migrate its collection of PDF formatted ETDs to an XML-based system using the ETD Document Type Definition (DTD). Even though the library lacks broad campus buy-in for its ETD project and only two departments have requirements to submit electronically as well as in print form, the library is using its ETD database of metadata with links to PDF document files (32 full text documents) as one of several document collections that will be implemented under a software architecture that is XML-based. This change to an XML-based architecture will advance the library's over arching digital initiative because of XML's powerful and flexible structuring capabilities, its ability to bridge different systems, and its capacity to capture and organize metadata. XML is a richer, more robust markup language that offers an effective solution for making available the intellectual content of collections.

A brief overview of current practice frames questions about the need to use XML and is followed by a more detailed view of aspects of XML that impact system design (DTDs and XML Schemas, database storage issues moving from file system to a relational database, and indexing issues using SQL Server 2000). The report concludes with a discussion of efforts to locate tools to convert existing PDF files into XML (software that allows direct, automated conversion of source files with XML tagged output for chapters, sections, paragraphs, figures, illustrations), and thoughts on the impact the change to an XML based system has on current staffing and workflow?

Background
The library joined NDLTD in 1998 and initiated an ETD pilot program in collaboration with the Graduate School. From the beginning the feeling was that to mandate electronic submission would be too difficult, particularly as the library staff with the support of the Graduate School was driving the impetus for the project. From the library's perspective, an ETD project offered a number of venues for pursuing digital library objectives and broadening staff experience, and continues to do so.

During the first year, pilot phase we developed and put in place policies and procedures for submitting electronic theses and dissertations, trained a student to convert documents to the PDF format and create metadata in an Access Database, and designed a web interface for public access that provided searching and browsing capabilities with retrieval of full-text PDF documents. The resulting ETD database was used to promote the project to the faculty and students. It was clear we needed faculty buy-in to effectively evolve towards a wholly electronic ETD service. As we moved out of the pilot phase two departments, Computer Science and Spatial Information and Engineering agreed to make delivery of an electronic thesis or dissertation mandatory.

Progress has been slow and we still do not have a requirement that all theses and dissertations be submitted electronically as well as in printed form. To demonstrate the merits of the project and expand awareness about it, in 2001 we began scanning all incoming theses and dissertations before the Library shipped them to the bindery, and we began to retrospectively scan existing theses and dissertations based on number of requests. Metadata (MARC catalog records) for the existing print collection documents was imported from the online catalog, parsed and loaded into the Access database. (Patrons can also access the full text documents through links in the online catalog). About this time we decided we were bumping up against the limitations of MS Access and wanted to expand our search capabilities. We migrated the data to MS SQL Server 2000, and implemented ranked searches using SQL Server's full text indexes. The web interface was redesigned and an interface for data entry developed and implemented for Special Collections staff that maintains the collection.

An alternative to PDF (which seems to have acquired the status of the de facto storage format for theses and dissertations) has always been a concern because we wanted a more stable and ongoing, standard format, one that was non-proprietary. We also wanted a format that would afford better search and retrieval capabilities with full text documents, better control of metadata, and one that would provide a solution for contributing metadata to the NDLTD central repository to aid in resource discovery. The Open Archives Initiative (OAI) is a constant on our radar screen, although we do not yet support the OAI protocol at Maine.

The current project team consists of: a graduate student (digital library assistantship), consulting programmer, systems administrator, head of special collections, reference librarian who has spearheaded our ETD project and an undergraduate student who has been responsible for document conversion.

Deciding to migrate from PDF to XML

The following reports documents the work of the project team to date, work in progress to migrate our collection of PDF formatted theses and dissertations to an XML-based system using the ETD DTD with modifications. Our current database (metadata for 2046 theses and dissertations with links to 32 PDF text files) is one of several pilot document collections that we anticipate implementing under a software architecture that is XML-based. The other collection uses the EAD DTD to encode finding aids for the papers of the former Secretary of State (William Cohen). In the larger digital library context one of our goals is to determine if it is possible to use XML as the foundation for a unified system for managing digital objects of different types with existing library staff. We would like to advance our overall systems management to a higher, more sustainable and open level, (a next stage system), one that systematically incorporates XML encoded metadata, thereby improving the global accessibility of our resources.

Our operating assumption is that changing to an XML-based architecture will advance our overall digital library initiative because of XML's powerful and flexible structuring capabilities, its ability to bridge different systems, its capacity to capture and organize metadata, and its capacity to preserve a document and its coding from obsolescence. Put simply, we felt that XML offers a more sophisticated way to find what you are looking for in a collection based on the structure and content of information in the collection. Although a new awareness of the power of XML as a standard is just surfacing among librarians, current usage is most prevalent in e-commerce. At this time, there are very few NDLTD collections using XML. However, it is an increasingly accepted, non-proprietary standard (W3 recommendation, 1998) for a rich, robust markup language that appears to be an effective solution for making available the "deep" intellectual content of a library's collections to a global community.

More particularly:

1. XML documents are easier to search than PDF documents because ultimately the XML document is simple text, which is understood by almost every application (as opposed to the proprietary binary format of a PDF).
2. PDFs are monolithic and can only be delivered to a browser as a whole, whereas chunks or sections of an XML document can be delivered even over low-bandwidth channels.
3. XML documents are easier to manage in a document collection because the metadata is embedded as an integral part of a document.
4. XML divides a document into logical parts that can be repurposed.

What is XML?

Understanding the essential difference between XML and PDF can be simply put: XML is all about structure and meaning, and has little to do with appearance; PDF is all about appearance, and has little or nothing to do with structure and meaning. They are complementary technologies. XML, Extensible Markup Language, focuses on the structure and content of information. Text or database elements are "fielded" – assigned tags that identify the kind of information they contain, and how different elements interrelate. The following section focuses briefly on some of the defining characteristics of XML integral to system design decisions. If the goal is to build a repository of information that can be used and adapted to new applications over time, XML provides advantages that are otherwise unavailable with PDF.

- **XML is a new industry standard that is non proprietary**
  PDF is a proprietary, de facto standard. XML is a true, independent standard (published as W3C recommendation in 1998).

  Because tagging facilitates intelligent searching and enables flexible redefinition of appearance, XML is rapidly being adopted for publishing on the web, replacing HTML and SGML. Because it is platform and media independent, it is also becoming the new standard for other forms of traditional and electronic publication, such as SGML.

  While XML may not yet have the support that HTML, PDF and SGML do, there appears to be industry support for it as companies like Microsoft, Adobe Acrobat, Arbor Text and Sun Microsystems are developing supporting applications and tools. One of the promises of XML is that cheaper and easier tools will be made available, thereby making XML easier and its use more widespread. The ideal workflow would start with XML, from which HTML and PDF can be generated.

- **Separates structure from presentation**
  XML (like SGML from which it is derived) delineates the elements of a textural document without describing appearance. Structural features of a document are separated from their presentation, which, ideally, is left to other applications to render. Basic text characters comprise both the text and the markup that describes the text. There are no proprietary codes

- **Markup language provides for greater accessibility to the intellectual content**
  XML, (like SGML)is a 'metalanguage', a language with which to make up and define markup systems. As a language XML does not offer a "standard" set of tags like HTML, but rather it provides a way to define a markup system for a particular document or class of documents (i.e. theses and dissertations). Standard HTML tags are concerned only with appearance and are limited to the set defined by the current standard. With XML you must define your own tags. XML tags mark structure and content rather than appearance; for

example, they can be used to mark document elements such as citations, appendices, bibliography, etc. as well as metadata elements such as thesis committee members, department and so on.

- **Offers both an XML DTD and an XML Schema**
  A note about the structure of an XML document: XML documents cannot exist independently as standalone documents. Because arbitrary tags define the elements on an XML document, every XML document must include a reference to another document where the tags and the relationships between tagged elements are defined. In early versions of the XML standards, only one definition standard was provided: the Document Type Definition or DTD. Current XML standards provide two parallel definition standards: DTDs and XML Schema.

  A DTD provides the framework for producing *well-formed* XML documents, and can be used to verify that documents are well formed. However, the DTD does not provide any mechanism for determining if a document is *valid.* Valid means that documents are well formed and also conform to a set of production rules. A DTD provides a name only for an element; XML schema provides both a name and a type. For example, in a DTD it is possible to define an element <defenseDate>; in XML Schema it is possible to indicate that <defenseDate> must be a valid representation of a date.

  In addition to type, XML Schema also supports the concept of a *namespace.* A namespace is a reference to a location where a set of names are defined within the same context. For example, many different documents might include an element called <department>, but corporations, department stores and universities all have departments. Within the namespace provided by an XML schema used in an academic document, a department might have an attribute such as Chair, while in a corporate environment a department might have an attribute such as Manager.

- **XML files include metatdata**
  Another significant advantage of XML is its ability to capture and organize information about information. XML is an excellent technology for managing metadata in a flexible and dynamic way, a fact born out by the proliferation of emerging metadata standards, e.g. CrossRef which expedites linking from one publisher's publications to another's.

- **Easy document display - XML Stylesheets (XSLT)**
  Because XML deals with content and not presentation, a separate document called a stylesheet is required for display. Extensible Stylesheet Language (XSL) was developed to meet these needs. (XSL consists of three parts: XSL Transformation (XSLT); XML Path Language (XPath) and XML Formatting Objects.) XSLT documents can be used to provide a set of rules under which an XML document can be translated into HTML on-the-fly. MathML is another tool for rendering mathematical equations.

System Design Issues:
(1) XML and a relational database structure

With the adoption of XML one of the first issues that needs to be addressed is system architecture. XML documents are usually thought of like any other document: a single electronic file that exists in a specific location. However, XML documents, like HTML documents, can be constructed on the fly so that the "document" only exists as a transient stream of bytes. Dynamic HTML documents are normally used for expressing content that changes with time.

Although ETDs are in fact static documents, there are two major reasons why it is desirable to store the documents in a format other than XML and generate XML representations dynamically: First, file-system based collections of documents are difficult to maintain and manage (what we have now). Second, file-system based collections of documents are difficult to search efficiently.

Alternatively, an XML document can be mapped to tables and columns in a relational database using a schema to design and build the storage representation of a collection. This design takes advantage of efficient document management and searching provided by relational database management systems (RDBMS) such as Oracle or SQL Server, in particular, services that store and access full text indexes and metadata and allow for the delivery of ranked queries.

SQL Server 2000 provides a powerful ability to support XML. We chose to continue development using SQL Server 2000 as our XML server, and to design a database based on both the ETD DTD and an XML Schema. Because an XML Schema provides both structure and type information, it can be used to define the storage representation of a collection of documents in a relational database. The XML Schema is not a replacement for a DTD; it supplements what is already in the DTD. Our plan is to use an XML-Data Reduced (XDR) schema (a subset of our XML Schema) to map the database to an XML view. SQL Server 2000 provides a number of annotations that can be used to map the XDR schema to the database tables and columns. Queries (Xpath) can be specified against the XML view created by the XDR schema to query the database and return results in the form of an XML document on the fly.

Ultimately the ETD system will probably contain the following components:

- The DTD (referenced by each document) provides structure to applications understanding DTDs.
- Full XML schema (referenced by each document) provides structure to applications understanding XML Schema.
- XDR Schema used internally by DB engine to generate documents on the fly.
- Documents

(2) The roles of the ETD DTD and the XML Schema

We examined the SGML DTD for the ETD-ML1.1 developed at Virginia Tech, adopted the evolving standard, and converted it to an XML DTD. In the conversion process we added tags to delineate such structural elements as: Table of Contents, List of Pictures, or Tables, or Figures, and added an “id” attribute to some tags, such as etd bibliography and vita.

XML Schemas are relatively new, while DTDs have been around for over a decade. The W3C XML Schema Definition Language is an XML language for describing and constraining the content of XML documents and W3C seems to be grooming XML Schemas as a replacement for DTDs, but it is not yet clear how quickly this transition will be made. We used the XML DTD to define our ETD XML Schema. Schema offers the following advantages to the system design:

(1) Schema support for data types which makes it easier to describe document contents, validate data correctness, work with data in a database, define data formats define data restrictions and convert data among different types;

(2) Schemas are written in XML which means you do not have to learn another language;

(3) Schemas because they are written in XML are extensible which means the schema can be reused in other schemata.

We chose to use an XML schema to describe the ETD structure, to design the database structure, and to build an XDR Schema to satisfy the needs of the SQL Server.

Populating the ETD database: Conversion from PDF to XML

We have a retrospective collection of 32 PDF documents that we want to transform into XML, and more urgently, we need to determine how we are to deal with documents to be received in the future. Ideally we would receive documents tagged in XML corresponding to our ETD Schema. However, none of the word processors currently in vogue provide XML support; and it would be impossible to require students to use obscure tools to manually apply XML tagging. Therefore manual intervention by library staff is required. We need to address the following questions:

1. How to convert the existing collection?
2. What format is best for conversion of future documents?

It is for all practical purposes virtually impossible to machine-recognize all document elements present in a thesis with currently available, off-the-shelf software. For example, available OCR (optical character recognition) software cannot distinguish structure in a document: a list of figures from a list of tables, or an abstract from a preface from an acknowledgment. The question is not whether manual editing is required but rather how much is required.

Clearly the conversion tool has to be carefully tailored to the particular structure that exists in the source PDF. Since our ETDs shared a consistent appearance and simpler layout, we looked for an off-the-shelf conversion tool that would render our XML documents with a minimum of manual intervention (post processing cleanup).

Conversion from PDF using Adobe Acrobat

In Acrobat 5 Adobe introduced a concept called "tagged PDF" which allows a software tool to add to its PDF writer some level of structure information to the PDF document, so that it contains more than just page layout information. Acrobat's offers two options to convert a PDF document to an XML document that would be based on the ETD DTD.

The first option is to use the default tree structure and convert the PDF to a tagged PDF. The tagged PDF can then be modified so that its tagging conforms to the ETD DTD. The complexity of the source PDF dictates the scale of mapping and modification that might ensue. Unfortunately, we could find no way to alter the default logic tree used to create a tagged PDF, which would reduce the magnitude of XML editing in cleanup.

The second option with Acrobat is to ignore the default logic tree and create from a blank template the ETD DTD, manually defining the tagging structure to be used. Using the "new child element" the user enters valid tags for the document in hand. Then converts the document to XML.

Conversion from RTF using Majix

Rich Text Format (RTF) is an early attempt at developing a markup language that would allow highly formatted documents to exist in a "text-only" format. Most popular word processors provide a decent level of RTF support. Because RTF itself is a markup language, RTF formatted documents are somewhat easier to convert to XML/XSL than are PDF documents, where the document elements have to be machine-recognized before being tagged.

With Majix "you can automatically transform RTF files in XML." It also allows for the user to customize the software to fit a locally defined tag set. The conversion from PDF to RTF to XML eliminated all tag definition in the document and labeled every line as paragraph <p>. Using the same software we tried converting from Word format to RTF to XML. The results were a little better, with some paragraph recognition.

I guess the good news is that once there is widespread support for a 'tagged PDF', the task of converting PDF to XML might become easier. The bad news is we are not quite there yet. We can only hope that in the near future XML support will be added to popular word processing software.

Conclusion

The XML based system design using SQL Server 2000 has reasonable potential for implementation, if not immediately for the ETD collection of PDF files, for the other pilot collection of EAD encoded finding aids. We do not yet have a resolution to the problem of conversion software. Since the number of documents is low (32), we may use the Adobe software and trained student labor to encode the documents manually. This approach addresses the retrospective collection but fails to solve the problem of ongoing database maintenance. We are exploring the development of a user interface to mask the complexity of the data entry. Lastly, the change from PDF to XML represents significant change in library workflow and staff responsible for the system design and maintenance of the collections. Staff in the relevant positions require technical skills or tools to hide the technical complexity.