XML for ETDs

By:

Aniket Prabhune and Dr. Edward Fox
{aprabhun, fox}@vt.edu
Department of Computer Science
Virginia Polytechnic Institute and State University

Abstract

The main objective of this project was to devise a tool/procedure to aid students at Virginia Tech in developing their electronic theses and dissertations (ETDs) in eXtensible Markup Language (XML) and to document properly all the work that was done at Virginia Tech in this regard. The project began by studying the other ETD-XML projects done earlier. Both the approaches (DTD and XSD) explored at Virginia Tech were studied and an attempt was made to improve the XSD approach using VBA (Visual Basic for Applications). The proposed approach was completely implemented and documented in a way that should be easy for the students to comprehend. This should help ease student efforts to prepare theses in XML.

Table of Contents

1. Background of the project

1.1 What is an ETD?

An ETD is a document that explains the research or scholarship of a graduate student. It is expressed in a form simultaneously suitable for machine archives and worldwide retrieval. The ETD is similar to its paper predecessor. It has figures, tables, footnotes, and references. It has a title page with the authors' name, the official name of the university, the degree sought, and the names of the committee members. It documents the author's years of academic commitment. It describes why the work was done, how the research relates to previous work as recorded in the literature, the research methods used, the results, an interpretation and discussion of the results, and a summary with conclusions.

The ETD is different, however. It provides a technologically advanced medium for expressing your ideas. You prepare your ETD using nearly any word processor or document preparation system, incorporating relevant multimedia objects, without the requirement to submit multiple copies on 50 percent cotton bond paper. Consequently, ETDs are less expensive to prepare, consume virtually no library shelf space, and never collect dust. At your choice, they can be available to anyone that can browse the World Wide Web.

1.2 What is the purpose of preparing an ETD?

By preparing an ETD and submitting it electronically you learn about electronic document preparation and about digital libraries. These skills will help prepare you for your future role in the Information Age, whether you teach, research, or use the research results of others.

Furthermore, you may be able to better convey the message of your thesis or dissertation in an electronic as opposed to a paper document. Thus, you can easily have color diagrams, color images, hypertext links, and even include audio, video, animations, spreadsheets, databases, simulations, virtual reality worlds, etc. in your appendices.

By submitting electronically you also allow your university to fulfill more economically its responsibilities of recording and archiving your thesis or dissertation. This is a key responsibility of the university, that is easier and less costly to fulfill when the workflow involves electronic documents.

Furthermore, paper documents can easily be produced from electronic documents, but not vice versa.

By not accepting paper, the University reduces handling and library costs, saves you money, and makes it possible for access to increase.

The website http://www.ndltd.org/ (Networked Digital Library of Theses and Dissertations) provides information about the history, description and scope of the ETD initiative. According to this website the main goals of the ETD initiative are:

- For graduate students to learn about electronic publishing and digital libraries, applying that knowledge as they engage in their research and build and submit their own ETD,

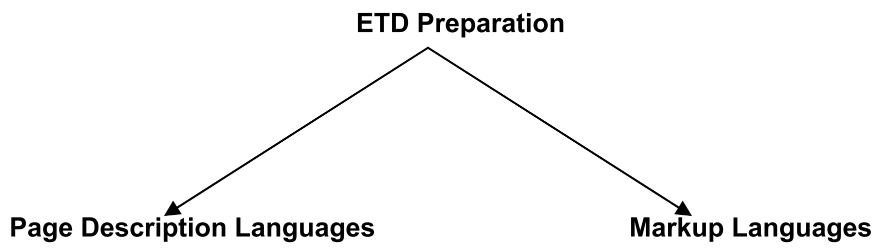
- For universities to learn about digital libraries, as they collect, catalog, archive, and make ETDs accessible to scholars worldwide,

- For universities in the southeast and beyond to learn how to unlock the potential of their intellectual property and productions,

- For graduate education to improve through more effective sharing, and

- For technology and knowledge sharing to speed up, as graduate research results become more readily and more completely available.

1.3 ETD preparation



1.3.1 Page description languages

The most popular approach to preparing ETDs is to a use a word processing system like MS Word, Corel Word Perfect, Star Office Word package, etc., that involves vendor specific, proprietary schemes or systems like Tex/LaTex, the Unix suite of troff, tbl, eqn, etc. Page description languages can then be used to represent documents created by such word processing systems.

A page description technique that is widely used for producing ETDs is Adobe's Portable Document Format (PDF). Adobe also provides tools for creating, annotating and manipulating PDF documents. A student can convert his thesis or dissertation into PDF from word processor systems in a matter of minutes by printing his document to Adobe's distiller printer driver.

PDF is the most widely supported format for ETDs and is a standard currently used at Virginia Tech. For more instructions on PDF and preparing ETDs in PDF you can visit http://etd.vt.edu and go to the guidelines section.

However, since ETDs should be usable across time and space, it is desirable that access to them is through suitable interchange formats, rather than transient, unpublished representations produced by particular versions of word processing systems or page description languages.

1.3.2 Markup languages

SGML (Standard Generalized Markup Language) and **XML** (eXtensible Markup Language) are markup languages, which use tags (“<” and “>”) with names of labels inside, around the sections of the documents that are thus marked or bracketed. A Document Type Definition (DTD) specifies the grammar or structure for a type or a class of documents. SGML requires a DTD while XML employs DTDs optionally.

1.4 Why XML?

The website http://www.w3.org/XML/ highlights various characteristics of XML which are also applicable to the suitability of XML for ETDs.

XML is a method for putting structured data in a text file.

For "structured data" think of such things as spreadsheets, address books, configuration parameters, financial transactions, technical drawings, etc. Programs that produce such data often also store it on disk, for which they can use either a binary format or a text format. The latter allows you, if necessary, to look at the data without the program that produced it. XML is a set of rules, guidelines, conventions, for designing text formats for such data, in a way that produces files that are easy to generate and read (by a computer), that are unambiguous, and that avoid common pitfalls, such as lack of extensibility, lack of support for internationalization/localization, and platform-dependency.

XML looks a bit like HTML but isn't HTML.

Like HTML, XML makes use of *tags* and *attributes* (of the form name="value"), but while HTML specifies what each tag and attribute means (and often how the text between them will look in a browser), XML uses the tags only to delimit pieces of data, and leaves the interpretation of the data completely to the application that reads it. In other words, if you see "<p>" in an XML file, don't assume it is a paragraph. Depending on the context, it may be a price, a parameter, a person, or something else. In short it allows you to develop your own mark-up language specific to a particular domain.

In addition, there are specific rules for XML that are stricter than the usual HTML markup. These strict rules are based on a notion of what is called well-formedness. Well-formedness is a new concept introduced by XML, which is not present in HTML. Essentially this means that all elements must either have closing tags or be written in a special form (as described below), and that all the elements must nest. The following five points are derived from the World Wide Web Consortium's (W3C) web site http://www.w3.org/TR/xhtml1/.

1) Although overlapping is widely tolerated in existing browsers, it cannot be done in XML.

- CORRECT- nested elements:
  <p>here is an emphasized <emph>paragraph.</emph></p>
- INCORRECT- overlapping elements:
  <p>here is an emphasized <emph>paragraph.</p></emph>

2) Element and attribute names must be case sensitive in XML, e.g., <li> and <LI> are different tags.

3) For non-empty elements (see #5 below for explanation of an empty element), end tags are required. In HTML, certain elements were permitted to omit the end tag; with the elements that followed implying closure. This omission is not permitted in XML.

- CORRECT- terminated elements:
  `<p>here is a paragraph.</p><p>here is another paragraph.</p>`
- INCORRECT- unterminated elements:
  `<p>here is a paragraph.<p>here is another paragraph.`

4) Attribute values for elements/tags must always be quoted, that is, contained within single or double quote marks (whichever you choose, either " or ' you must match them for each attribute; you cannot have "something', but instead, "something" or 'something').

For instance there is a common tag, the anchor or "a"-tag used to make a link, where the element you use is "a" and there are the attributes of "href" for where the link goes to:

```
<a href="http://www.vt.edu/" class="link">my link</a>
```

So, for working with attributes or "variables" which are part of an element or tag, you should be careful to note this.

- CORRECT- quoted attribute values:
  `<table rows="3">`
- INCORRECT- unquoted attribute values:
  `<table rows=3>`

5) Empty elements must either have an end tag or the start tag must end with />. For instance, <hr /> **NOT** <hr> (the tag for horizontal rules):

- CORRECT- terminated empty tags (note space before the "/"):
  ```
  <br />
  <hr />
  ```
- INCORRECT- unterminated empty tags:
  ```
  <br>
  <hr>
  ```

XML documents can be preserved for a long time.

XML is, at a basic level, an incredibly simple data format. It can be written in 100 percent pure ASCII text as well as in a few other well-defined formats, e.g., Unicode. ASCII text is reasonably resistant to obsolescence. Also XML is very well documented. The W3C's XML 1.0 specification tells us exactly how to read XML data.

XML is license-free, platform-independent, and well-supported.

By choosing XML as the basis for some project, you buy into a large and growing community of tools (one of which may already do what you need!) and engineers experienced in the technology. Opting for XML is a bit like choosing SQL for databases: you still have to build your own database and your own programs/procedures that manipulate it, but there are many tools available and many people that can help you. And since XML, as a W3C technology, is license-free, you can build your own software around it without paying anybody anything. The large and growing support means that you are not tied to a single vendor. XML isn't always the best solution, but it is always worth considering.

XML is a family of technologies.

There is XML 1.0, the specification that defines what "tags" and "attributes" are, but around XML 1.0, there is a growing set of optional modules that provide sets of tags & attributes, or guidelines for specific tasks. There is, e.g., *Xlink* which describes a standard way to add hyperlinks to an XML file. *XPointer* & *XFragments* are syntaxes for pointing to parts of an

XML document. (An Xpointer is a bit like a URL, but instead of pointing to documents on the Web, it points to pieces of data inside an XML file.) CSS, the style sheet language, is applicable to XML as it is to HTML. *XSL* is the advanced language for expressing style sheets. The *DOM* is a standard set of function calls for manipulating XML (and HTML) files from a programming language. *XML Namespaces* is a specification that describes how you can associate a URL with every single tag and attribute in an XML document. What that URL is used for is up to the application that reads the URL, though. *XML Schemas* help developers to precisely define their own XML-based formats. There are several more modules and tools available or under development.

XML can encode metadata about DTDs.

Documents are often supplemented with metadata (that is data about data). If such metadata were included inside an ETD then it would make that ETD self-describing. XML can encode such metadata.

XML adds additional requirements.

On the down side XML adds some additional requirements.

Conversion from word processing forms to XML requires more planning in advance, different tools, and broader learning about document processing concepts. Writing directly in XML by using XML authoring tools requires some prior knowledge of XML. Also XML is very strict regarding the naming and ordering of tags and is case sensitive. These factors make the conversion process complicated, difficult, and time consuming as compared to that for PDF.

1.5 Some basics of XML that we should know before we proceed

DTD (Document Type Definition):

An XML document consists of a strictly nested hierarchy of elements with a single root. Elements can contain character data, child elements, or a mixture of both. The *structure* of the XML document is described in the DTD. There are different kinds of documents like letter, poem, book, thesis, etc. Each document has its own structure. This specific structure is defined in a separate document called Document Type Definition (DTD). Whenever an XML document is associated with a DTD, its structure should conform to that defined in the DTD.

CSS (Cascaded Style Sheets):

CSS is a flexible, cross-platform, standards-based language used to suggest stylistic or presentational features applied throughout entire websites or web pages. In their most elegant forms, CSS are specified in a separate file and called from within the XML or HTML header area when documents load into the CSS-enabled browser. Users can always turn off the author's styles and apply their own or mix their important styles with the author’s. This points to the "cascading" aspect of CSS.

CSS is based on rules and style sheets. A **rule** is a statement about one stylistic aspect of one or more elements. A **style sheet** is one or more rules that apply to a markup document.

An example of a simple style sheet is a sheet that consists of one rule. In the following example, we add a color to all first-level headings (H1). Here's the line of code - the rule - that we add:

```
H1 {color: red}
```

XSD (XML Schema):

A schema is an abstract definition of an object's characteristics and interrelationships, i.e., it contains no actual data, only potential data. An XSD can be thought of as a blueprint for making and checking XML instances. It is less abstract than RDF (Resource Description Framework), but provides deeper control and growth than DTDs. An XSD file is written in XML, thus is made up of a collection of elements and attributes. The file extension of .xsd is used to indicate that a file is an XML Schema rather than an XML instance. You can read an XSD file with an XML editor. XML Schema allows each element and attribute, content models as well as the data types that represent the actual information, to be defined. An XSD simply provides a method for defining an element (e.g., element name="...") or referring to one that already exists (e.g., element ref="...").

XSL (eXtensible Stylesheet Language):

XSL is a language for expressing style sheets. It consists of two parts:

1) A language for transforming XML documents, and
2) An XML vocabulary for specifying formatting semantics.

If you don't understand the meaning of this, think of XSL as a language that can transform XML into HTML, a language that can filter and sort XML data, a language that can address parts of an XML document, a language that can format XML data based on the data value (like displaying negative numbers in red), and a language that can output XML data to different devices, like screen, paper, or voice. XSL is developed by the W3C XSL Working Group whose charter is to develop the next version of XSL.

Because XML does not use predefined tags (we can use any tags we want), the meanings of these tags are not understood: <table> could mean an HTML table or maybe a piece of furniture. Because of the nature of XML, the browser does not know how to display an XML document.

In order to display XML documents, it is necessary to have a mechanism to describe how the document should be displayed. One of these mechanisms is CSS as discussed above, but XSL is the preferred style sheet language of XML, and XSL is far more sophisticated and powerful than the CSS used with HTML.

XML Namespaces

The purpose of XML namespaces is to distinguish between duplicate element type and attribute names. Such duplication might occur, for example, in an XSLT style sheet or in a document that contains element types and attributes from two different DTDs. An XML namespace is a collection of element type and attribute names. The namespace is identified by a unique name, which is a URI. Thus, any element type or attribute name in an XML namespace can be uniquely identified by a two-part name: the name of its XML namespace and its local name. This two-part naming system is the only function of XML namespaces.

- XML namespaces are declared with an xmlns attribute, which can associate a prefix with the namespace. The declaration is in scope for the element containing the attribute and all its descendants. For example, the code below declares two XML namespaces. Their scope is the A and B elements:

```
<A xmlns:foo="http://www.foo.org/" xmlns="http://www.bar.org/">
   <B>abcd</B>
</A>
```

- If an XML namespace declaration contains a prefix, you refer to element type and attribute names in that namespace with the prefix. For example, code below declares A and B in the http://www.foo.org namespace, which is associated with the *foo* prefix:

```
<foo:A xmlns:foo="http://www.foo.org/">
  <foo:B>abcd</foo:B>
</foo:A>
```

- If an XML namespace declaration does not contain a prefix, the namespace is the default XML namespace and you refer to element type names in that namespace without a prefix. For example, code below is the same as the previous example but uses a default namespace instead of foo prefix:

```
<A xmlns="http://www.foo.org/">
  <B>abcd<B>
</A>
```

2 Approaches to ETD production in XML

Basically there are two approaches that are used for producing ETDs in XML.

The first approach is to use a template, whereby a user is provided with a template and fills in the required details.

The second approach is to use conversion software that would take a Word or RTF document as input and provide the user with its XML equivalent.

2.1 Template approach

The template approach allows end users to create their full text in a way which results in an easier eventual conversion process, without requiring them to understand fully the markup process. However, the template approach requires more end-user training, and requires authors to create their work in one of a limited set of applications (e.g., Word and LaTex) for which templates are supported and have been created.

In the template approach the user is provided with the template of tags in which he is supposed to enter the relevant details. This approach suits best for front and back matter of ETDs since their content mostly remains the same.

A part of the template for the front matter may look as follows. The user needs to enter the relevant details for the body and the back matter template in the same way. For full template of front, body, and back matter please refer to Appendix B. It has the complete DTD used in the template approach project led by Tejas Patel at Virginia Tech in Spring 2001.

```
<?xml version="1.0"?>
<?xml:stylesheet type="text/css" href="test2.css" ?>
<! DOCTYPE ETD SYSTEM "ETD-LATEST.DTD">

<FRONT>
```

```
<TITLEPAGE>
     <DOCTITLE>
        Put the title of your thesis here
     </DOCTITLE>

         <DOCAUTHOR>
          Put the name of the author of the thesis here
         </DOCAUTHOR>

         <WORKTITLE>
           Put the purpose of the document here
         </WORKTITLE>

         <SUBJECT>
          Put the subject of your thesis here
         </SUBJECT>

         <THESISADVISOR>
               Put the name of your thesis advisor here
         </THESISADVISOR>

         <DATE>
          Put the date here
         </DATE>

         <LOCATION>
          Enter the location of the school here.
         </LOCATION>

         <KEYWORDS>
          Enter keywords used to identify this document here.
        </KEYWORDS>

         <THESISCOPYRIGHT>
          Enter copyright information here.
         </THESISCOPYRIGHT>

   </TITLEPAGE>

     <ABSTRACT>
         Abstract goes here
     </ABSTRACT>
   ……..
   ……..
   ……..
   </FRONT>
```

Each of these tags shown in the template has been predefined in the DTD and their formatting style is present in a CSS file. Each and every XML file would require some version of the top three lines to be present since they carry specific meaning.

- The first line <? xml version = “1.0” ?> is a processing instruction and specifies the version of the XML specification used.
- The second line <!DOCTYPE ETD SYSTEM “etd-latest.dtd”> identifies the root element and indicates that an external DTD (etd-latest.dtd) is being associated with the XML document. The root element is the one of which all other elements are subset. In our case the root element is ETD.

- The third line <?xml: stylesheet type = “text/css” href = “test2.css”> identifies the style sheet file that we are going to use for this document. This cascaded style sheet contains all the formatting information.

Once the user fills in all the required information in the front, body and back matter template, then he needs to save the file with a different file name with extension .xml. After doing this, the user would have his XML file ready and he can fire up an XML compatible browser and see the results.

2.2 Conversion tool approach

The conversion approach takes as its input one or more acceptable formats that end-users are accustomed to (Word or LaTex for example). Most if not all of the tools available to convert electronic texts into XML rely on people (e.g., the students themselves) trained to identify and mark up structural (and even presentational) elements. The tools used for this are for the most part text editors. Even in cases where some conversion can be performed automatically (i.e., interpreting structure from style information), the author must proofread the computer-generated output carefully.

Once the full text has been marked up (by whichever means), there remains a third common stage: making use of the XML file in some way. This can be as simple as converting the XML to a display format (HTML, PDF) using XSLT and XSL/FO (XSL/Formatting Objects), but can also include extracting structure and data to build tables of contents, citation databases, metadata indexes, etc.

The trickiest part of the conversion tool approach is the conversion part, where the user is supposed to convert from one format to XML format. There are tools available that allow us to convert from one format to XML, but all are not well suited to our particular needs. Most of the tools use their predefined DTD and CSS files and the user needs to be smart enough to format the document according to his needs. One of the tools that we tried at Virginia Tech was MAJIX. MAJIX converts an RTF document to an XML document. Below is the short description of the process that is used to convert a RTF document to XML document using XML.

NOTE: MAJIX is very limited in its conversion approach and is not recommended for most of the applications. It is explained here to give you an idea of how the conversion approach works.

1) The first step is to convert the existing Word document into RTF format. This is very simple and can be done within Word by choosing File, Save As, then under Save File as Type, choose Rich Text Format (*.rtf).
2) Start MAJIX--assuming it was installed correctly--by double clicking the “majix_jre.bat" shortcut on the desktop.
3) Once the program is running, you will see a window with two white blanks and several buttons at the bottom. Click on the "**Find File**" button, and choose your RTF file from your test directory. MAJIX will automatically rename the output to something with ".xml" as the extension. Then click the bottom left button "**Convert**" and wait.
4) Now open your newly-made XML file in Notepad or WordPad and you will see something like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- generated by Majix from C:\WINDOWS\Profiles\strabala\Desktop\Majniri\Manjiri
chapter2.rtf on Wed May 31 13:27:03 CDT 2000 using template MyDoc -->
<!DOCTYPE mydoc PUBLIC "-//TetraSix//DTD mydoc v1//EN" "mydoc.dtd" [
<!ENTITY tab >
]>
<mydoc>
<info>
<title>Chapter 2: Quartiers</title>
<author>Tejas Patel</author>
<operator>Patty </operator>
<company> </company>
</info>
```

You must delete the above information from your .xml file and substitute the following information:

```
<?xml version="1.0"?>
<?xml:stylesheet type="text/css"  href="test2.css " ?>
<!DOCTYPE ETD SYSTEM "etd-latest.dtd">
<MATTER>
<SECTION>
<SUBSECTION>
```

That's all there is to making this basic conversion. You can even read this document in a browser now. There are many other tools in the market that allows you to convert from some form to XML. However none of them is specifically designed to support conversion of ETDs.

3 Virginia Tech approaches to ETD preparation in XML:

There are several methods for controlling and validating XML files using DTD (Document Type Definition), XML-Schema (XSD), and Resource Description Framework (RDF) files. These controls are embodied as files with appropriate extensions, e.g., .dtd, .xsd, or .rdf. Virginia Tech has explored the approaches for validating the XML files, using DTD and XSD.

3.1 Procedure 1 to create an ETD in XML at Virginia Tech using DTDs

3.1.1 Format of ETDs used for XML conversion

To facilitate proper maintenance of ETDs, an ETD was divided into three logical parts as shown in the list below.

1 Front portion
2 Body (Matter) portion
3 Back portion

The DTD that was prepared at Virginia Tech for the ETDs also reflects the same structure and has various tags associated with each part as discussed below.

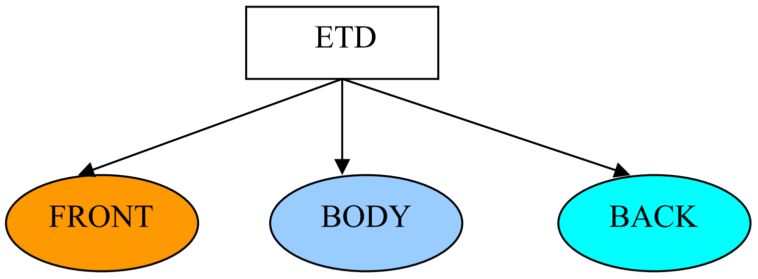


Figure 1: DTD structure for ETDs

3.1.1.1 Front portion

It contains information about the thesis or dissertation. It includes TITLEPAGE, ACKNOWLEDGEMNTS, ABSTRACT, TOC, PREFACE, etc. The front portion, more than any other area, has the most specific set of tags.

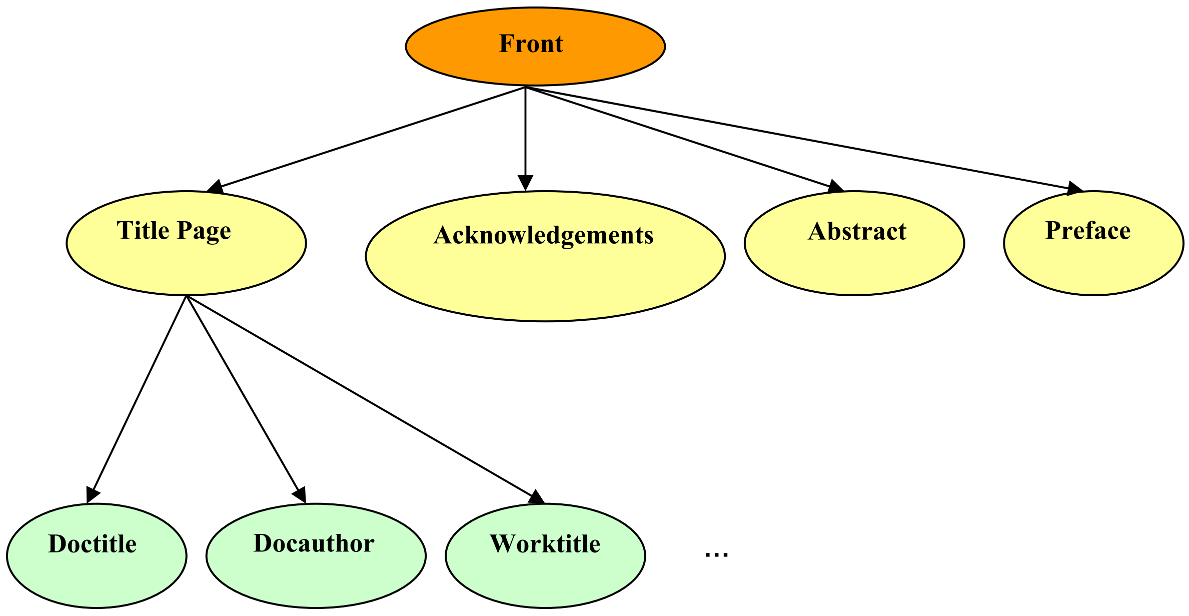


Figure 2: Front portion of the DTD for ETDs

Components of front portion:

The following table shows the tags that are found only in the front portion. These are displayed according to which tags belong to which "parent" tags, with "FRONT" being parent of all. There are more tags than these, but these are those that should not be used elsewhere than in the "FRONT" portion. Those with a question mark "?" are optional. The “+” mark means that at least one occurrence is needed and “*” indicates that there must be zero or more occurrences. The tags are shown in order from left to right and top to bottom.

FRONT	TITLEPAGE	DOCTITLE+
		DOCAUTHOR+
		WORKTITLE*
		SUBJECT
		THESISADVISOR+
		DATE
		LOCATION
		KEYWORDS
		THESISCOPYRIGHT
	ACKNOWLEDGEMENTS?	
	ABSTRACT	
	TOC?	
	PREFACE?	

Table 1: Tag structure for the front portion of DTD for ETDs

3.1.1.2 Body (Matter) portion

This is the main part of the ETD that contains the content of the ETD. This normally would include one or more chapters. However we do not have an explicit element called “CHAPTER” in the DTD. A chapter is represented as a chapter title and one or more sections that in turn would contain subsections and paragraphs. Sections and subsections can be included recursively. A typical body portion would look as follows. Note that the chapter element is shown below just to clarify the concept.

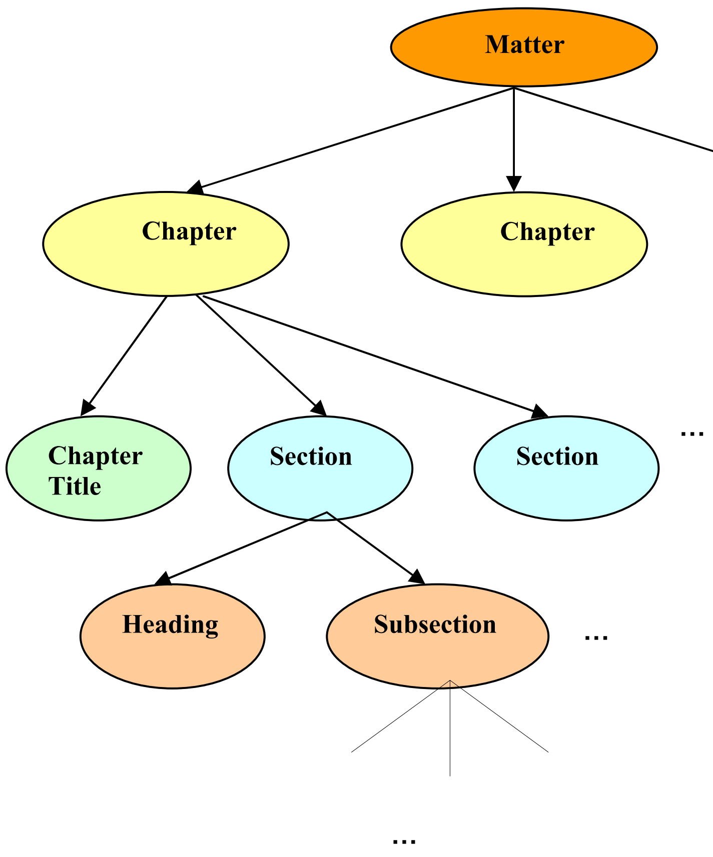


Figure 3: Body portion of the DTD for ETDs

Components of body portion:

The following table describes the main tags of the Body (Matter) portion. The body portion always begins with the “MATTER” tag. Those with a question mark "?" are optional. The “+” mark means that at least one occurrence is needed and “*” indicates that there must be zero or more occurrences. The tags are shown in order from left to right and top to bottom.

	SECTION+	HEADING*		….
		SUBSECTION*	HEADING*	
			img*	
			Link (a)*	
			LIST*	
			table*	
		img*		
		Link (a)*		
		LIST*	ITEM+	
		table	tr (row)*	
			td (cell)*	
			th*	
MATTER	CHAPTERTITLE*			

Table 2: Tag structure for the body portion of DTD for ETDs

Note: We do not have an element like Chapter, but we do have sections and subsections, which can be called recursively.

3.1.1.3 Back Portion

This would contain the end portion of an ETD. This typically includes bibliography, appendices, citations, and vita.

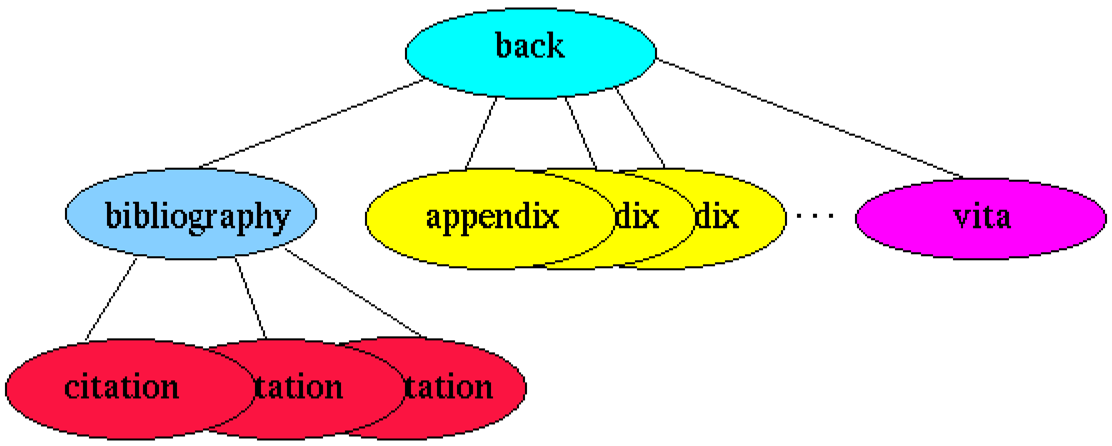


Figure 4: Back portion of the DTD for ETDs

Components of the back portion:

The following table describes the main tags of the back portion. The back portion always begins with the “BACK” tag. Those with a question mark "?" are optional. The “+” mark means that at least one occurrence is needed and “*” indicates that there must be zero or more occurrences. The tags are shown in order from left to right and top to bottom.

BACK	BIBLIOGRAPHY
	APPENDIX*
	CITATION?
	PUBLISHER*

Table 3: Tag structure for the back portion of DTD for ETDs

The DTD used in this approach is given in Appendix C.

3.1.2 Approach

NOTE: The thesis document used here as sample may have been truncated and may not fully represent original content.

1) Here is the basic type of information for inclusion in the header of each and every document you create in XML for your ETD:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="test2.css"?>
<!DOCTYPE ETD SYSTEM "ETD-LATEST.DTD">
<ETD xmlns:xhtml='http://www.w3.org/1999/xhtml
xmlns:html='http://www.w3.org/TR/REC-html40>
```

- The top line will tell browsers that document is XML as well as how to read any special language characters.
- XML allows you to specify different character set encoding. The encoding must be identified within the <?xml ?> declaration as an encoding="UTF-8" attribute. An XML processor is required to support 'UTF-8' and 'UTF-16'. If you are working in a Non-Roman-Script language such as Japanese, use UTF-16 as the value for the "encoding" attribute in the <? xml ?> declaration.
- The third line with "DOCTYPE" identifies the basic root element (that element of which all others are a subset), which for Virginia Tech ETDs is "ETD" (remember that XML is case-sensitive).
- Next comes the namespace definition. We need to define the XHTML and HTML namespaces that allow us to use some standard HTML tags in our ETD. This also forms the starting tag of our document and every document should then be terminated with an </ETD> tag.

2) The next step is to define one of the three parts of the document. If you are creating the front portion of the document, then the next tag should be <FRONT>, if body part then <MATTER> and if you are creating the back part then use the <BACK> tag. So depending on your type of file, your file may look like one of the following.

a) Front portion

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="test2.css"?>
<!DOCTYPE ETD SYSTEM "ETD-LATEST.DTD">
<ETD xmlns:xhtml='http://www.w3.org/1999/xhtml' xmlns:html='http://www.w3.org/TR/REC-
html40'>

<FRONT>
    <TITLEPAGE>
    <DOCTITLE>
        NetEdit: A Collaborative Editor
  </DOCTITLE>

 <DOCAUTHOR>
        Ali Asghar Zafer
 </DOCAUTHOR>

<WORKTITLE>
    Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University in
    partial fulfillment of the requirements for the degree of Masters of Science in Computer
    Science
</WORKTITLE>

<THESISADVISOR>
        Clifford A. Shaffer,
        <ROLE> Chairman </ROLE>
 </THESISADVISOR>

<THESISADVISOR>
    Roger W. Ehrich,
    <ROLE>Co-Advisor</ROLE>
</THESISADVISOR>

<THESISADVISOR>
Manuel A. Perez-Quinones,
<ROLE>Co-Advisor</ROLE>
</THESISADVISOR>

<DATE>April 23, 2001</DATE>
<LOCATION>Blacksburg, Virginia</LOCATION>

<KEYWORDS>
Keywords: Computer-Supported Cooperative Work, Collaboration, Editor, and Replication
</KEYWORDS>

<THESISCOPYRIGHT>
Copyright 2001, Ali Asghar Zafer
</THESISCOPYRIGHT>
</TITLEPAGE>
<ABSTRACT>
```

```
    Put your abstract here
</ABSTRACT>
<ACKNOWLEDGEMENTS>
Put your Acknowledgements here
</ACKNOWLEDGEMENTS>
</FRONT>
</ETD>
```

b) Body portion

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="test2.css"?>
<!DOCTYPE ETD SYSTEM "ETD-LATEST.DTD">
<ETD xmlns:xhtml='http://www.w3.org/1999/xhtml' xmlns:html='http://www.w3.org/TR/REC-
html40'>
<MATTER>
<CHAPTERTITLE>
<html:a name="1">1.Introduction </html:a>
</CHAPTERTITLE>
<SECTION>
<SUBSECTION>
We have been through various phases of Information technology from centralized mainframe
computing to desktop systems and more recently distributed computing. With the explosive
growth of Internet and pervasive use of computers, we need to think about shifting from personal
computing to inter-personal computing. Since people perform a lot of tasks working together, and,
because a computer is used in much of their work, it is interesting to explore ways to use
computers to allow this collaboration. With organizations getting global and the cost of travel for
face-to-face collaboration getting higher and time consuming, it is becoming more appealing to
research computer based collaboration………..
</SUBSECTION>
<SUBSECTION>
<HEADING>
<html:a name="1.1">1.1Problem Description </html:a>
</HEADING>
This work describes a collaborative editor; technical issues in its design and implementation, and
its usability study. A collaborative editor allows two or more users to remotely edit a document
simultaneously. The editing is completely unconstrained and the users can insert and delete
characters at any location; in fact two or more users could be performing insertions and deletions
at exactly the same position...…..
</SUBSECTION>
        ….
</SECTION>
</MATTER>
</ETD>
```

c) Back portion

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="test2.css"?>
<!DOCTYPE ETD SYSTEM "ETD-LATEST.DTD">
<ETD xmlns:xhtml='http://www.w3.org/1999/xhtml' xmlns:html='http://www.w3.org/TR/REC-
html40'>

<BACK>
```

<BIBLIOGRAPHY>
<HEADING>
References
</HEADING>
<LIST>
<ITEM>
1. Gutwin, C., Greenberg, S., &amp; Roseman, M., "Workspace Awareness in Real-Time Distributed Groupware: Framework, Widgets, and Evaluation", People and Computers XI (Proceedings of the HCI'96), Springer-Verlag, Pages 281-298.
</ITEM>
< ITEM >
2. Fish, Robert S., Kraut, Robert E., Leland Mary D. P., and Cohen, M., "Quilt – A Collaborative Tool for Cooperative Writing", Proc. COIS’88 Office Information Systems (Palo Alto, CA, March 1988).
</ ITEM >
< ITEM>
3. Leland, Mary, Fish, Robert and Kraut, Robert, "Collaborative Document Production Using Quilt", Proc. CSCW ’88 Computer-Supported Cooperative Work (Portland, Or., September 1988).
</ ITEM >
</LIST>
</BIBLIOGRAPHY>
<APPENDIX>
<HEADING>
Appendix A
</HEADING>
<SUBSECTION>
<HEADING>
A.1 Installation Instructions
</HEADING>
The following are the installation instructions for downloading and setting up the environment for executing NetEdit.
<HEADING>
Installing the Server:
</HEADING>
<LIST>
<ITEM>
1.First download the server side files from the following URL:
<html:a href=” http://csgrad.cs.vt.edu/~azafer/thesis/thesisPage.html” target=”_blank”>
http://csgrad.cs.vt.edu/~azafer/thesis/thesisPage.html.
</html:a>
There will be a link to the .zip file at the top of the page.
</ ITEM>
< ITEM >
2. Create a new directory named NetEditServer under the root directory say C:\
</ ITEM>
<ITEM>
3. Extract the contents of the .zip file in the directory created in step 2.
</ITEM>
< ITEM>
4. If you do not have java runtime environment installed, then go to the following URL:
<html:a href=http://www.java.sun.com/j2se/1.3/download-windows.html target=”_blank”>
http://www.java.sun.com/j2se/1.3/download-windows.html
</html:a>
Download Java 2 SDK, v 1.3.0 Software for Windows 95/98/2000/NT 4.0 as a one large bundle. Then install this environment by double clicking on the downloaded file and following the instructions. Select the default directory and drive to install the program.

</ITEM>
<ITEM>
5. Go to the directory NetEditServer created in step 2 and give the following command from the console:
Here Server address corresponds to the IP address of the machine on which the above steps are being performed. The port number can be any free port (say 2500).
<html:br/>
e.g., java ServerLogin thyme.cs.vt.edu 2500
<html:br/>
To add a user to the system, add a line to the PasswordFile.txt. This file is present in the directory created in step 2. The format of this line is as follows:
<html:br/>
</ITEM>
</LIST>
</SUBSECTION>
</APPENDIX>
</BACK>
</ETD>

3.1.3 Explanation of key tags and their usage

1) Our DTD has been formatted in such a way that you can use standard HTML tags since we incorporate the XHTML and HTML namespaces. So to use any HTML related tag the common syntax is
   Starting tag: <html: (tag name)>
   Closing tag: </html: (tag name)>

2) For certain empty tags of HTML like <br>, <hr> the syntax to be used is
   <html:br/> or <html:hr/>

3) In order to incorporate links in your documents we can use the standard "anchor" tag of HTML.
   Usage:
   <html:a href="www.vt.edu " target="_blank":> Link Name here</html:a>
   One can also use the standard attributes of the anchor tag like target, which specifies in which window the new link opens. Default is the same window. "href" attribute is compulsory.

4) In order to incorporate images in your document, the procedure is the same as that in links. Use the standard "img" tag of HTML.
   Usage:
   <html:img src="vt.gif"> </html:img>

5) For incorporating tables in your document, proceed as follows.
   Usage:
   <html:table [attributes]>
   <html:tr>
   <html:td>Data here</html:td>
   </html:tr>
   </html:table>

6) For having a list of items, we have defined two tags in our DTD, LIST and ITEM.
   Usage:
   <LIST>

<ITEM>data</ITEM>
<ITEM> data </ITEM>
</LIST>

7) Several characters are part of the syntactic structure of XML and will not be interpreted in the usual fashion if simply placed within an XML document. You must substitute a special character sequence called a **predefined entity** as per the table below.

Reserved character	Predefined entity to use instead
<	&lt;
&	&amp;
>	&gt;
'	&apos;
"	&quot;

Table 4: Predefined entities for reserved characters in XML

3.2 Procedure 2 to create an ETD in XML at Virginia Tech using XML Schema

The XSD (XML Schema) was designed to validate the conformance of a student's XML document. A schema was used in place of a Document Type Definition (DTD) because of the limiting factors that DTDs imposed on the author of an XML document like:

- DTDs are poorly suited for using namespaces while XSDs use namespaces well.
- DTDs cannot differentiate between the data types, i.e., there is no way to differentiate between "10", the string and 10 the number thus requiring the applications that use XML to have an existing convention for denoting which fields are numbers and which are strings.
- A DTD is not written in XML meaning that validating parsers must be written to understand DTDs using separate capabilities to those they use for manipulating XML.
- A DTD does not have a concept of inheriting some or all of the characteristics of another DTD. This limits the extensibility of the DTDs.

To begin work on the schema, the Metadata schema from :
http://www.ndltd.org/standards/metadata/etdms/1.0/etdms.xsd was used in order to provide information for the "front portion" of the ETD.

In order to render the XML document easily in any web browser, a transformation language, XSLT (e**X**tensible **S**tyle **L**anguage **T**ransformation), is used. We chose to render our document in HTML because this allows for viewing the ETD on a common web browser. An XSL processor reads the ETD that is written in XML and described by the XSD file; when it finds a tag that it knows how to render it follows the template rules that are supplied in the XSL style sheet file and then it outputs an HTML document. A rule may specify that something is to be rendered in bold, or italics, or a special font. The author of the XSL style sheet file has the ability to choose how he/she wants a particular tag rendered. It makes the most sense to understand the context of what is being rendered when making decisions on how something is rendered. For example, a heading may be bolded and in larger text, or a title to an article may be italicized and in a normal font. If the defined XSD is extended in the future by defining additional tags, then these tags too can be rendered by extending the XSL style sheet to include the template rules for the additional tags. Diagrammatically this can be shown as below:

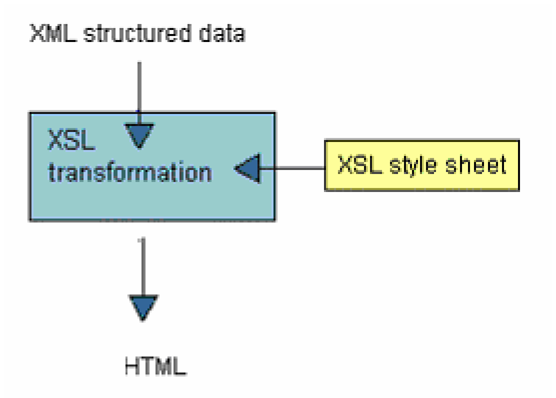


Figure 5: Rendering the XML document in HTML using XSLT

3.2.1 ETD creation – why smart tags failed

One of the goals of the project conducted by a student group of CS-5604 class in Fall 2001 was to find a relatively easy way for ETD authors to create or convert an ETD into our XML format. Originally, Microsoft Smart Tag technology was planned for this purpose. Smart Tags are supposed to be used in instances where we have specific or well-defined terms that require some sort of action. For example, it is easy to write a Smart Tag that looks for words like "Microsoft" or "Linux", and then opens up a certain website. It also is possible to look for specific terms stored in a database, say a list of names or such. It is even possible to look for non-specific terms as long as we define them sufficiently, such as "a 5-digit string" (a zip code).

The ETD tags that we use, however, are based on items that are far from that specific and well defined. When a line such as "The Industrial College of the Armed Forces: Contextual Analysis of an Evolving Mission" appears, it is not entirely clear what that is from the words alone. Based on its physical location and the formatting of the words, we can discern that it is the title to an ETD. The same situation occurs for names like "Scott Masterson" or "David Tessendorf". Smart Tags might be able to tell that these are names, but are they the creators of the ETD, or committee members, or what?

A few attempts at making smart tags work were made. A template was initially made for the ETD metadata. This allowed smart tags to easily figure out things like who was a creator and what was a title. However, making the technology work for the ETD body tags, such as <section> and <chapter>, was not successful when attempted by a student group in Fall 2001. At the root of the problem is the fact that smart tags can only work off paragraphs. Chapters and sections can span many paragraphs, and can be nested as well. It became obvious that smart tags were not meant for the type of ETD/XML creation that would be needed.

3.2.2 ETD creation – Visual Basic for Applications

There is another technology that does a much better job of allowing for the type of interaction needed. It is Visual Basic for Applications (VBA). Basically, it is the Visual Basic language embedded into Office applications, such as Microsoft Word. It acts as a sort of extended macro language, and enables developers to have a lot more control of a Word document.

Using VBA, all of the text parsing and conversion that occurred with smart tags can be replicated. Thus, a template for ETD metadata can be used, as before, and converted into XML text. Furthermore, VBA can check when a user has selected a block of text (or multiple paragraphs), and allow the selection to be wrapped with the XML tag of their choosing. Finally, VBA code can look for text in specific formats and process them accordingly. Thus, the user can "encode" their document using specific styles. For example, a "Section" style can be created and used throughout the ETD.

Basically, with VBA, the power of a real programming language lies behind our document. For users to create their XML ETD, they will be able to use a template Word document and just paste in their ETD text. They will be able to easily tag the content of their ETD simply by selecting the portions of text followed by a right click and then selecting the relevant tags from the macro options provided to them.

3.2.3 Format of ETDs used for XML conversion

To facilitate proper maintenance of ETDs, the ETD was divided into three parts as shown in the list below.

1 Front portion
2 Body portion
3 Back portion

The XSD that we prepared for the ETDs also reflects the same structure and has various tags associated with each part as discussed below.

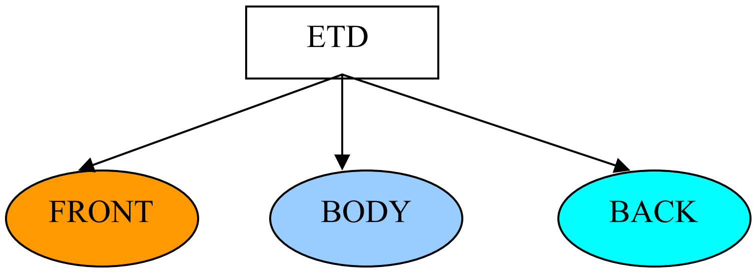


Figure 6: XSD structure for ETDs

3.2.3.1 Front portion

It contains information about the thesis or dissertation. It includes title, creator, subject, description, degree, name, acknowledgment, etc. The front portion, more than any other area, has the most specific set of tags.

Components of front portion:

The following table shows the tags, which are found only in the front portion. These are displayed according to which tags belong to which "parent" tags, with "front" being parent of all. Those with a question mark "?" are optional. The "+" mark means that at least one occurrence is needed and "*" indicates that there must be zero or more occurrences. The tags are shown in order from left to right and top to bottom.

front	title+			
	creator +			
	subject +			
	description *			
	contributor *			
	date			
	type +			
	format *			
	identifier +			
	language *			
	rights *			
	degree?	name *		
		level*		
		discipline *		
		grantor *		
	acknowledgement *			
	table?	label *		
		tablebody	row+	cell*

Table 5: Tag structure for the front portion of XSD for ETDs

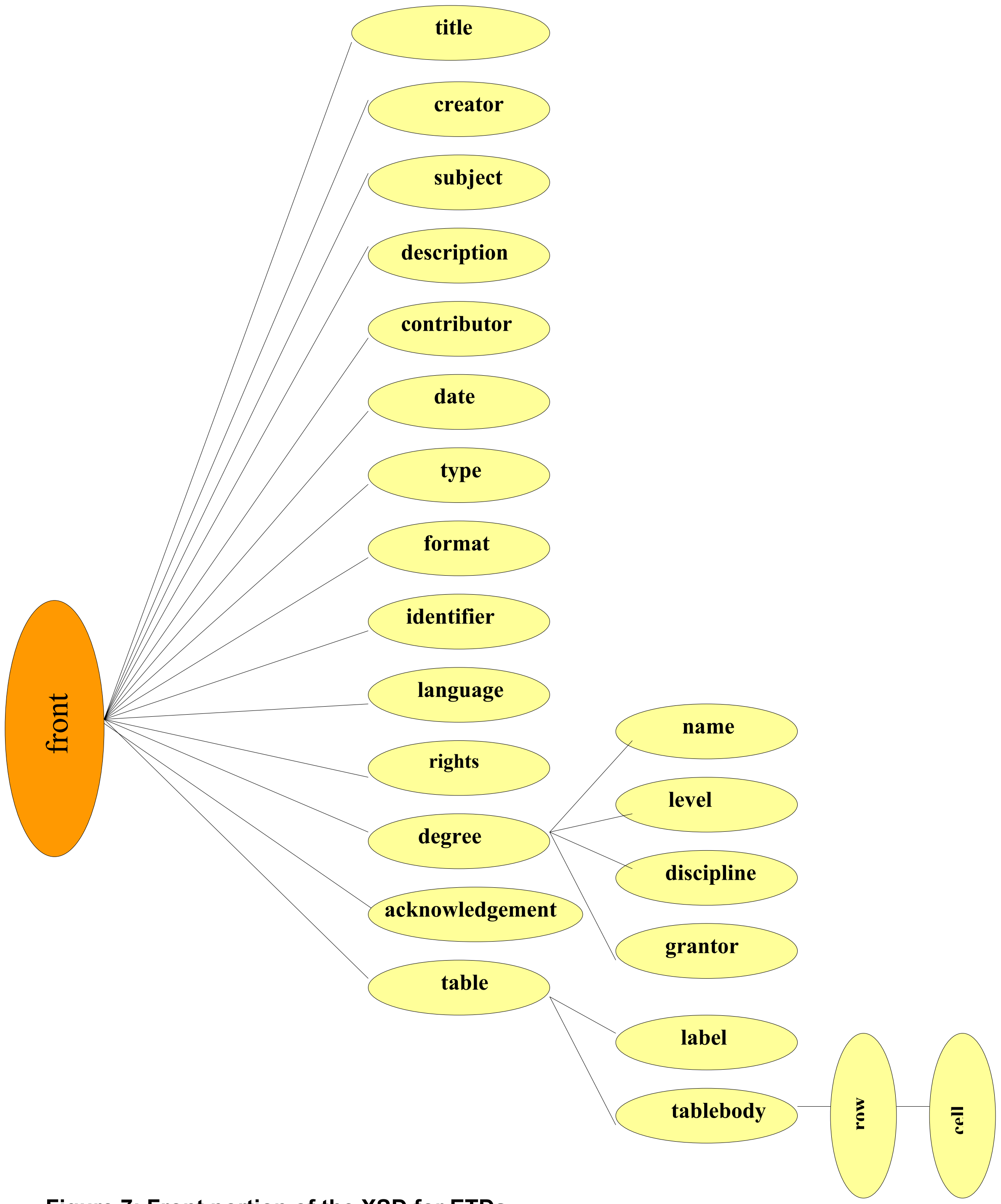


Figure 7: Front portion of the XSD for ETDs

3.2.3.2 Body portion

This is the main part of the ETD that contains the content of the ETD. This would include one or more chapters. Chapters would in turn contain chapter titles, sections, paragraphs, etc.

Main components of body portion:

The following table constitutes of the main tags of the Body Matter. The body matter always begins with the “body” tag. Those with a question mark "?" are optional. The “+” mark means that at least one occurrence is needed and “*” indicates that there must be zero or more occurrences. The tags are shown in order from left to right and top to bottom.

body	chapter*	chapterTitle				
		section+	section *			
			sectionTitle+			
			paragraph *			
			table *	label*		
				tablebody	row+	cell*
			link*			
			image *	label		
				filename		

Note: Section can be called recursively.

Table 8: Tag structure for the body portion of XSD for ETDs

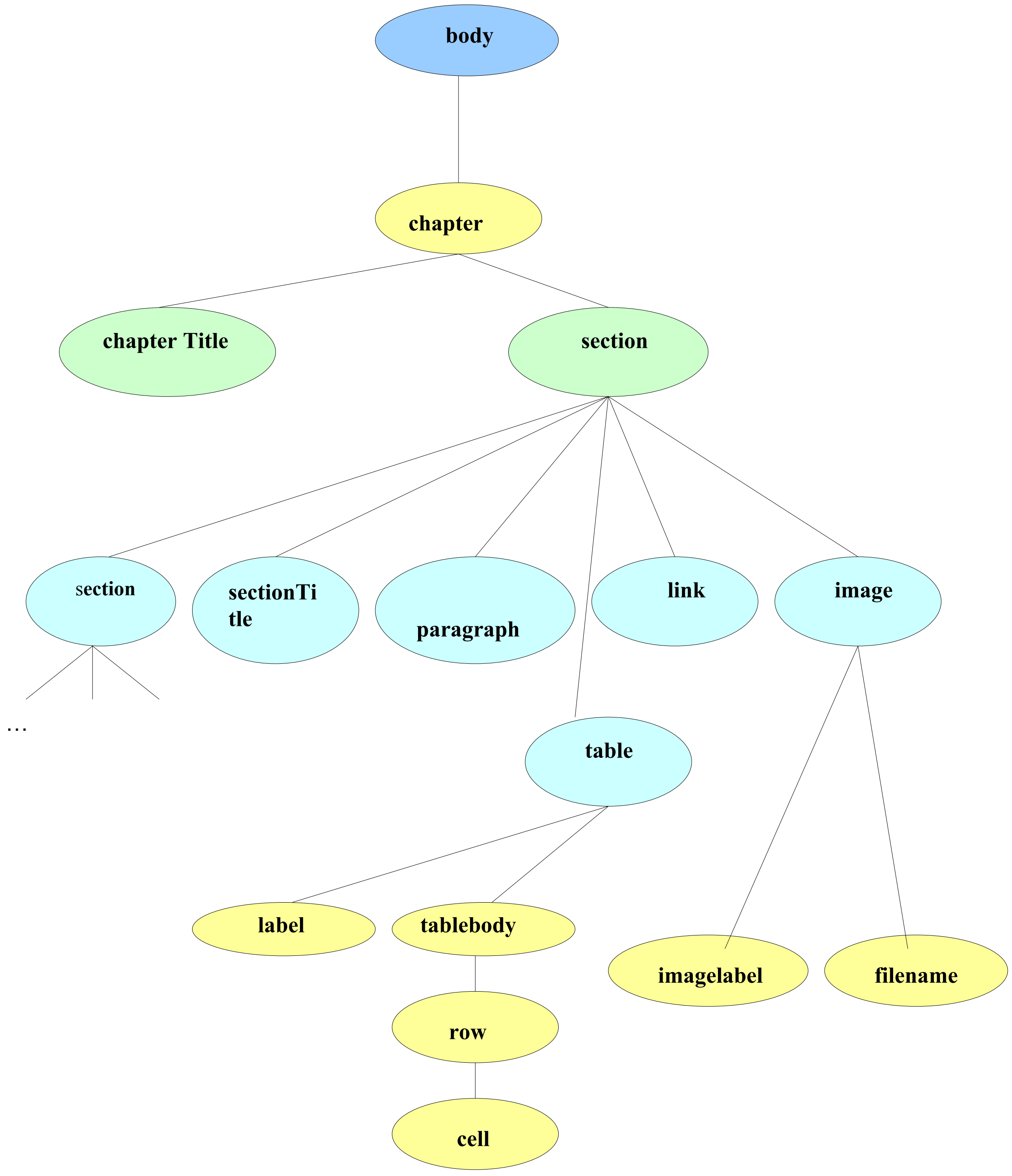


Figure 8: Body portion of the XSD for ETDs

3.2.3.3 Back portion

This would contain the end matter of an ETD. This typically includes bibliography, references, and appendices.

Components of the back matter:

The following table describes of the main tags of the Back Matter. The back matter always begins with the “back” tag. Those with a question mark "?" are optional. The “+” mark means that at least one occurrence is needed and “*” indicates that there must be zero or more occurrences. The tags are shown in order from left to right and top to bottom.

back	bibliography *	referenceTitle
		reference+
	appendix *	appendixTitle*
		section+
	vita*	vitaTitle*
		vitaAuthor*
		section*

Note: The section in the back portion has the same structure as described for the Body Matter.

Table 7: Tag structure for the back portion of DTD for ETDs

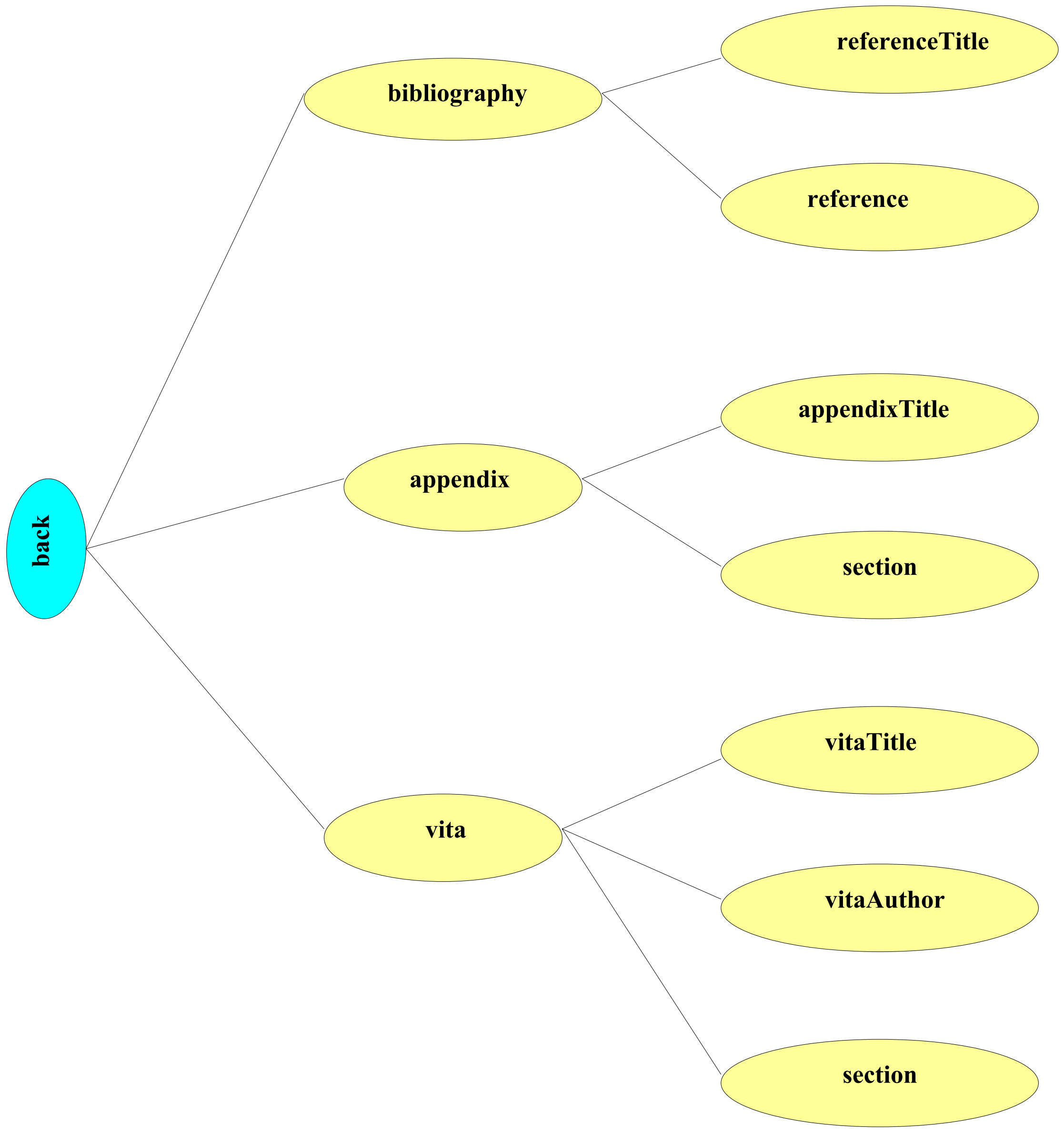


Figure 9: Back portion of the XSD for ETDs

APPENDIX A: XML for ETDs users guide

1. Create a folder for your work. For example: C:\thesis\
2. Get the following files in this folder from the website: http://www.ndltd.org/xml/etd/Imp-Files/
   a. XML Template.doc
   b. etd.xsl
   c. Sabcmd.exe
   d. Sablot.dll
   e. Expat.dll

   The files listed as c, d, e above can be obtained by uncompressing the sablot.zip file from the above website.
3. Open the XML Template.doc file using Microsoft Word 2000. Go to Tools→Macro→Security as shown below:

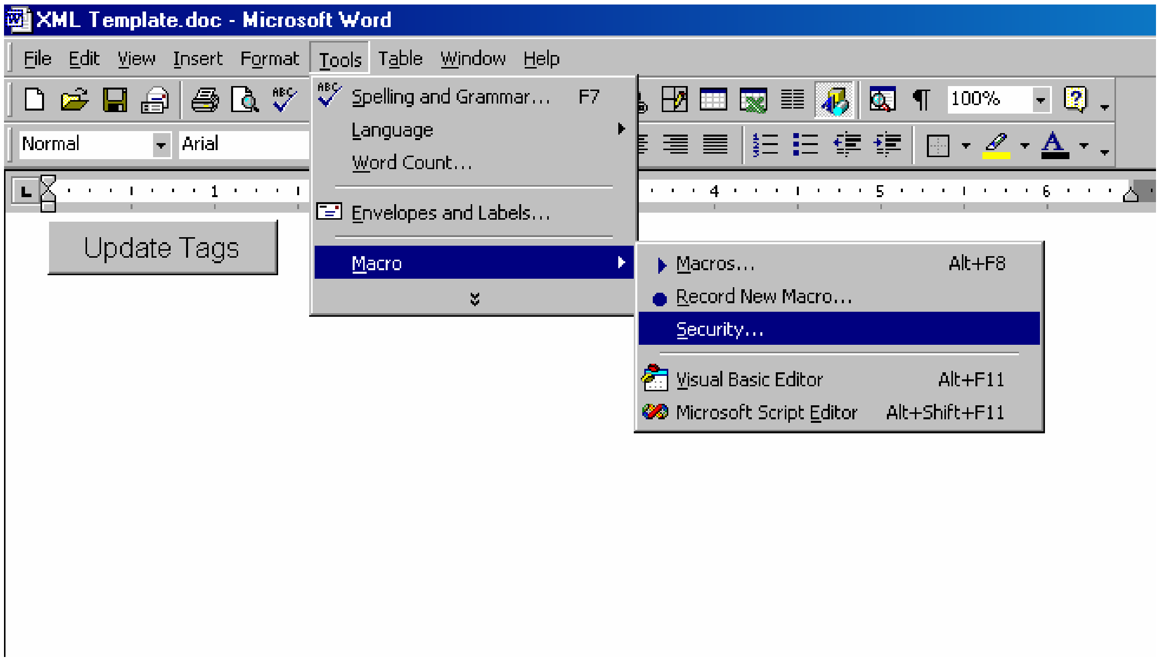
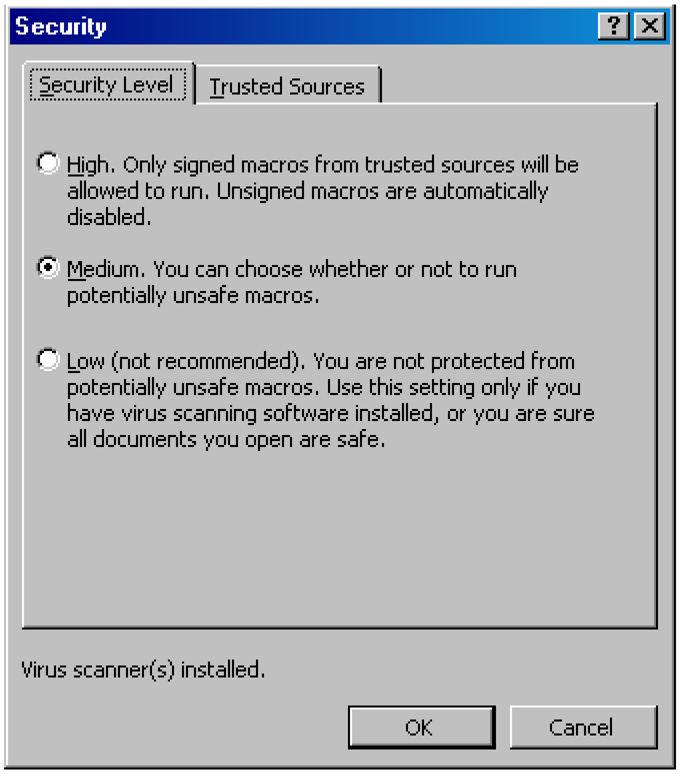

XML Template.doc - Microsoft Word
File Edit View Insert Format Tools Table Window Help
Normal
Arial
100%
Update Tags
Spelling and Grammar... F7
Language
Word Count...
Envelopes and Labels...
Macro
Macros... Alt+F8
Record New Macro...
Security...
Visual Basic Editor Alt+F11
Microsoft Script Editor Alt+Shift+F11

Figure 10: Macro security level option

This will show up a dialog box for the Macro Security Level as shown below:



High. Only signed macros from trusted sources will be allowed to run. Unsigned macros are automatically disabled.

[✓ Medium. You can choose whether or not to run potentially unsafe macros.]

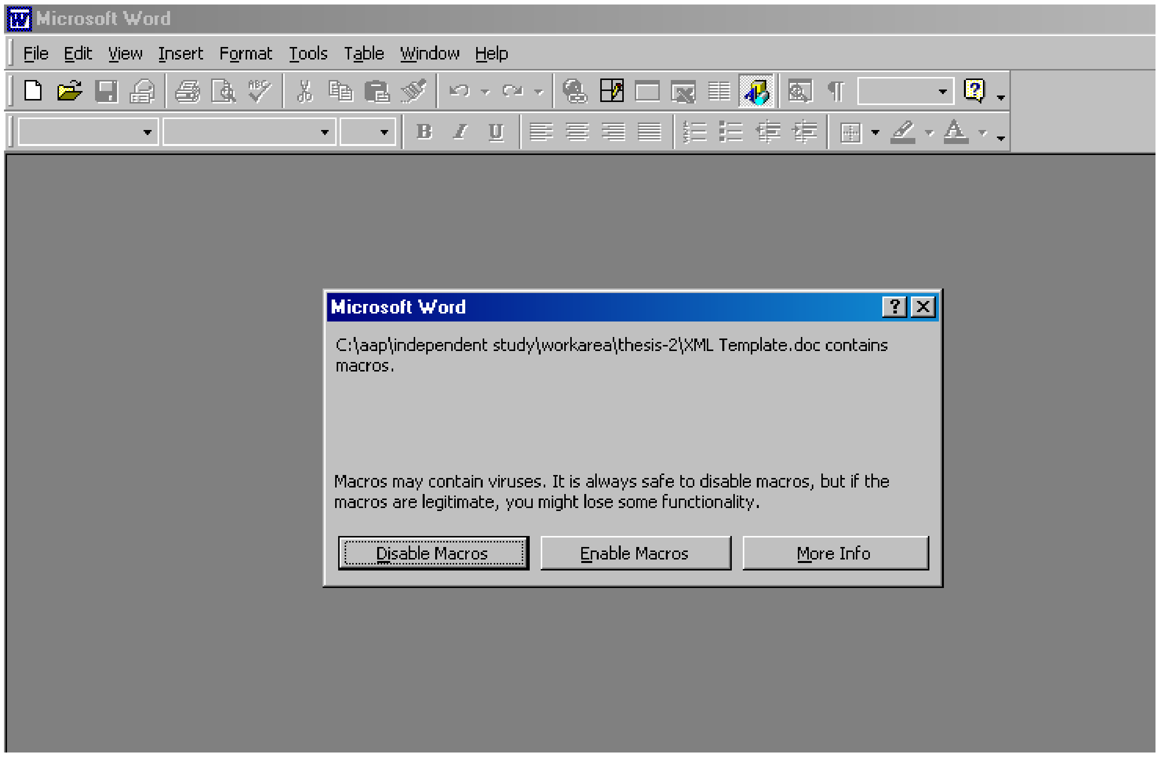
Low (not recommended). You are not protected from potentially unsafe macros. Use this setting only if you have virus scanning software installed, or you are sure all documents you open are safe.

Virus scanner(s) installed.

Figure 11: Macro security level dialog box

Ensure that the macro security level is set to "Medium".

4. Close the XML Template document and then reopen it. This is required whenever you change the macro security level, so that the new macro security level goes into effect. Opening a document with the macro security level "Medium" or "Low" leads to a dialog box giving a macro alert as shown below:



C:\aap\independent study\workarea\thesis-2\XML Template.doc contains macros.

Macros may contain viruses. It is always safe to disable macros, but if the macros are legitimate, you might lose some functionality.

Figure 12: Enable macros prompt

Click the Enable Macros button. This will enable the macros that are used for inserting the XML tag information.

5. Now copy your thesis text content from another PDF/Word document (if you had already created one) into the file XML Template.doc. However you can start creating the thesis in this document itself if you have not started yet. Note that you do not need to copy the image/multimedia files into the word document as these files will be kept separate, outside the XML document (in the work folder, e.g., C:\thesis\) and will be referred to within the XML document.

6. Follow a top down approach of inserting the tags into your document moving from the top of the tag structure and going into more and more detail. Tags can be easily inserted by selecting portions of text and then using right click to show a set of options for the various tags from which you should select the proper tag as shown below:

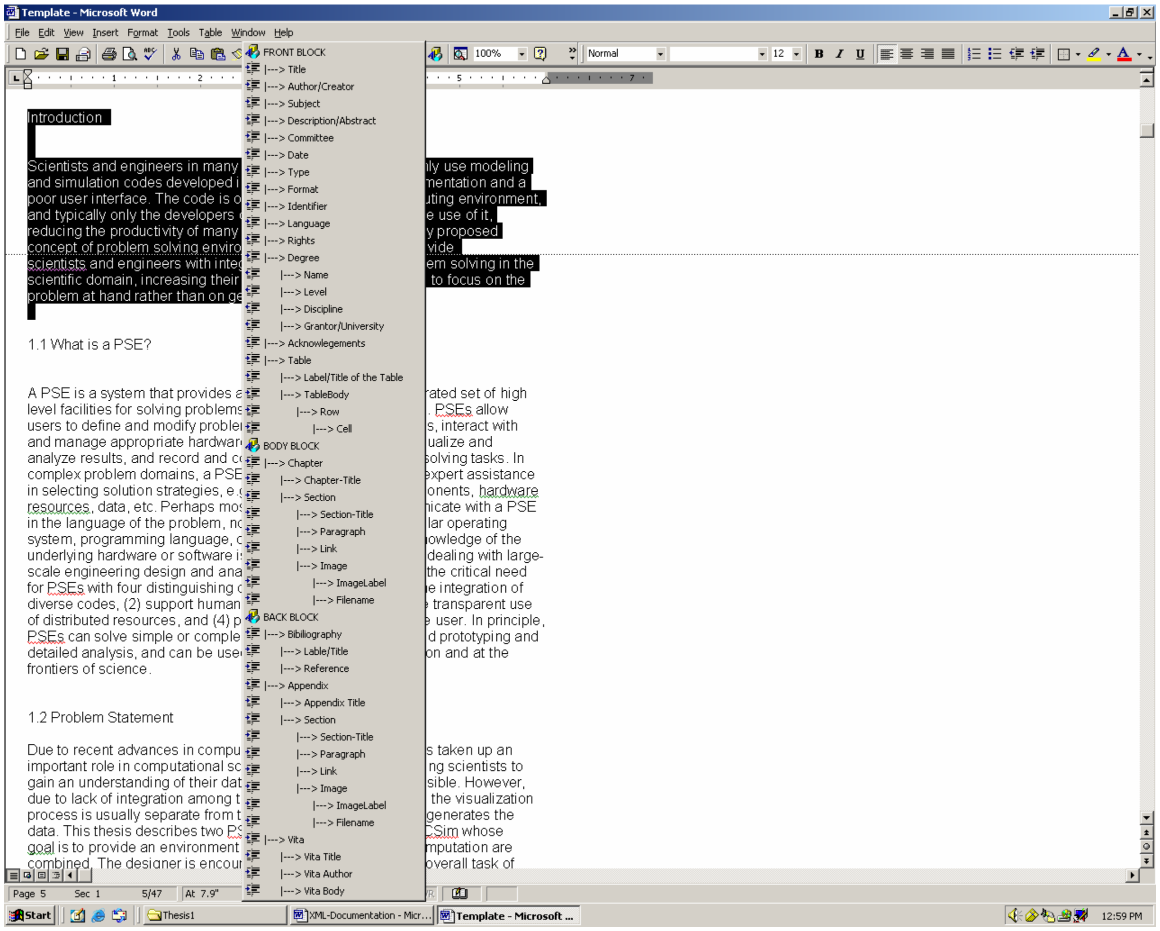


Figure 13: Macro tag options shown when the text is selected and right click is used

An explanation of the tag hierarchy structure and the procedure to insert the specific tags follows.

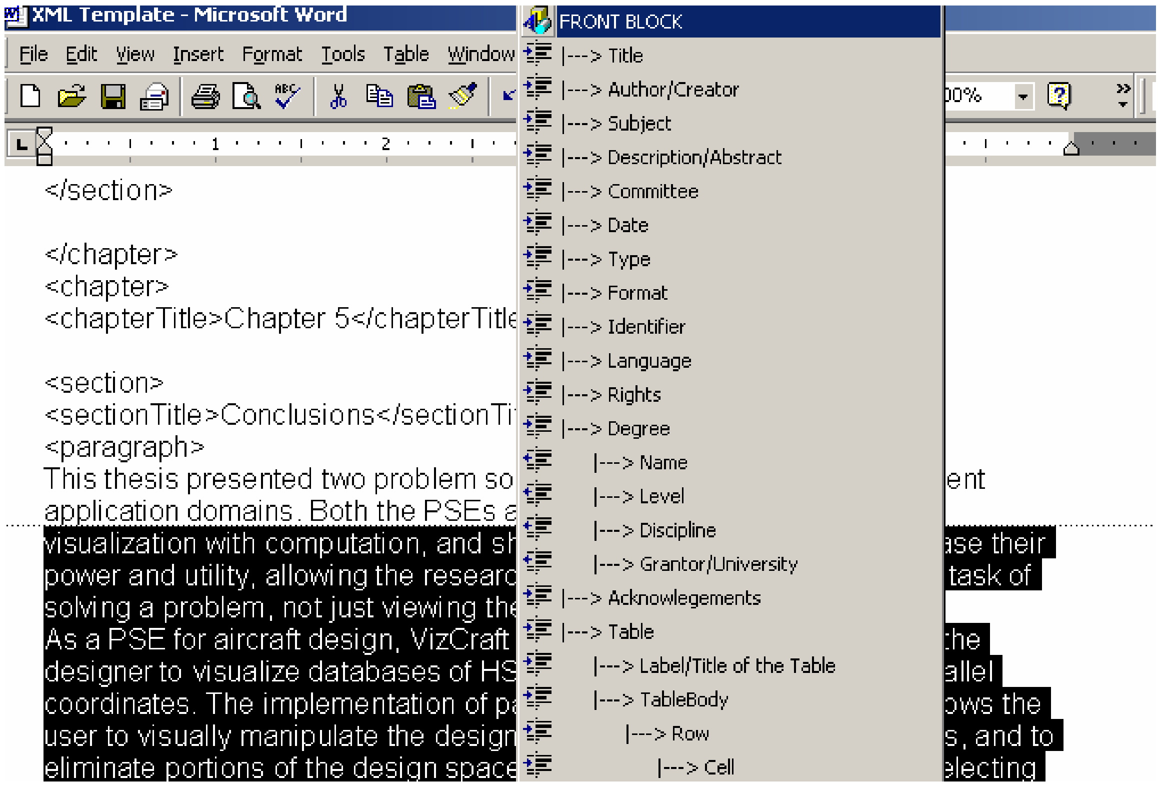
7. The entire thesis document is divided into 3 logical parts: the front, body, and back. Note that all these parts are within the same document. Divide the entire thesis into the above parts by selecting their respective portions (as given below), and tagging them (as explained in step 6) with the top level tags in the hierarchy of the tag structure (i.e., **front**, **body,** and **back)** and then within these parts tag the content according to the structure of each part.

   The **front** part includes all the descriptive introductory details about your thesis like:

1) Title
2) Author/Creator

3) Subject: Subject topic of the thesis.
4) Description: Abstract of your thesis.
5) Committee: Committee members of your thesis.
6) Date: A date associated with an event in the life cycle of the resource. In the case of theses and dissertations, this should be the date that appears on the title page or equivalent of the work.
7) Type: The nature or genre of the content of the resource. This field is used to distinguish the resource from works in other genres and to identify the types of content included in the resource. The string "Electronic Thesis or Dissertation" is recommended as one of the repeatable values for this element.
8) Format: Format in which the thesis is rendered, e.g., text/xml or text/html.
9) Identifier: Web address identifying your thesis on web.
10) Language: e.g., en (English).
11) Rights: Information about rights held in and over the resource. Typically, this describes the conditions under which the work may be distributed, reproduced, etc., how these conditions may change over time, and whom to contact regarding the copyright of the work.
12) Degree: This in-turn consists of:
    a. Name: e.g., MS, PhD
    b. Level: e.g., Masters, Doctorate
    c. Discipline: e.g., Computer Science
    d. Grantor: University issuing the degree: e.g., Virginia Tech
13) Acknowledgements: Acknowledgement statement.
14) Table: This in-turn consists of:
    a. Label/Title of the Table, e.g., “Table 1: Analysis data”
    b. Table body: This in-turn consists one or more of:
        i. Row: which in-turn consists of:
            1. Cell: These are the different columns within a row.

The tags for the front part can be inserted as explained in step 6. The options corresponding to the front part are as shown below:



```
</section>

</chapter>
<chapter>
<chapterTitle>Chapter 5</chapterTitle

<section>
<sectionTitle>Conclusions</sectionTi
<paragraph>
This thesis presented two problem so                    ent
application domains. Both the PSEs a
visualization with computation, and sh                  ase their
power and utility, allowing the researc                 task of
solving a problem, not just viewing the
As a PSE for aircraft design, VizCraft                  the
designer to visualize databases of HS                   allel
coordinates. The implementation of pa                   ows the
user to visually manipulate the design                  s, and to
eliminate portions of the design space                  electing
```

Figure 14: Macro tag options for the front portion

An example **front** portion of the thesis, after the tags have been inserted, will look like: (Note that the original contents of the thesis have been modified for brevity.)

<front>

<title>Information Literacy : A Study of Freshman Students' Perceptions, with Recommendations</title>

<creator>Nancy H. Seamans</creator>

<subject> Curriculum and Instruction (Instructional Technology) Philosophy</subject>

<contributor>David M. Moore, Chair</contributor>
<contributor>John K. Burton</contributor>
<contributor>Mary Ann Fitzgerald</contributor>
<contributor>Barbara B. Lockee</contributor>
<contributor>Jan K. Nespor</contributor>

<date>April 30, 2001</date>

<type>Electronic Thesis</type>

<format>text/xml</format>

<identifier>http://scholar.lib.vt.edu/theses/available/etd-061899-113642/</identifier>

<degree>

```
<name>Ph.D</name>
<level>Doctorate</level>
<discipline>Philosophy</discipline>
<grantor>Virginia Polytechnic Institute and State University</grantor>
</degree>

<description>
The research problem for this study is focused on the need to know how students acquire and
use information. Research indicates a lack of understanding of what students know about
information and how they use information and this study used the Information Literacy
Competency Standards for Higher Education (Appendix A) as the basis for acquiring a better
understanding of what kind of information freshman students at Virginia Tech need and how they
acquire it during their first semester at college. Students were asked questions about their
information use during fall semester 2000, using both email questioning and in face-to-face
interviews. The data collected was used to develop insights into how students acquire and use
information and resulted in suggestions that could be used in revising and improving instruction
for freshman students that is provided by the University Libraries at Virginia Tech.
</description>

<acknowledgement>
Sydney, Kathleen, Hal, Greta, Jeremiah, Mark, Eric, Roy, Damien, and Professor Robbins are the
heart and soul of this document; their time and their words were invaluable. Dr. Mike Moore's
feedback and encouragement helped shape my entire research effort, and his willingness to
spend exorbitant amounts of time guiding me through this process are greatly appreciated. Dr.
Rebecca Clark served as a mentor and as a patient ear, keeping me focused and on track. And
more than anyone else, for his support and encouragement, and willingness to participate in
interminable discussions and listen to much whining, I thank Geoff.
</acknowledgement>

<table>
<label>Table of Contents</label>

<tablebody>

<row><cell>i</cell><cell>Abstract</cell></row>
<row><cell>ii</cell><cell>Acknowledgements</cell></row>
<row><cell>iii</cell><cell>Table of Contents</cell></row>
<row><cell>1</cell><cell>Chapter 1 - Introduction and Review of Literature</cell></row>
        <row><cell>1.1 </cell><cell>Introduction</cell></row>
        <row><cell>1.2 </cell><cell>Review of Literature</cell></row>
<row><cell>1.3 </cell><cell>Summary</cell></row>
<row><cell>1.4 </cell><cell>Significance of the Literature</cell></row>
…

<row><cell>References</cell></row>
<row><cell>Appendix A  </cell><cell>Information Literacy Competency Standards</cell></row>
<row><cell>Appendix B  </cell><cell>Questions</cell></row>
<row><cell>Appendix C  </cell><cell>Institutional Review Board (IRB) information</cell></row>
<row><cell>Appendix D  </cell><cell>Demographic Data</cell></row>
<row><cell>Appendix E  </cell><cell>Coding Data</cell></row>
<row><cell>Appendix F  </cell><cell>Researcher Bias and Checks</cell></row>
<row><cell>Vita</cell></row>

</tablebody>
</table>
</front>
```

The **body** part includes all the main contents of your thesis in the form of various chapters like:

1) Chapter: This in turn consists of:
    a. Chapter-Title: This is the title of your chapter
    b. Section: The section involves the main contents in form of :
        i. Section: A section can in turn have other sections recursively.
        ii. Section-Title: This is the title of the section.
        iii. Paragraph : These are the text portions of your content.
        iv. Table: This in-turn consists of :
            1. Label/Title of the Table, e.g., “Table 1: Analysis data”
            2. Table body: This in-turn consists one or more of :
                a. Row: which in-turn consists of :
                    i. Cell: These are the different columns within a row.
        v. Link: This enables to add links in your document. Note that when you select a text and then tag it as Link a default form like this would appear (assuming the text you highlighted was “my-pic”):
        <simpleLink href=”filename”>my-pic</simpleLink>
        You then need to change the “filename” to the URL/URI you want to refer to, e.g., “photo1.jpg”.

        vi. Image: This is used in order to refer to images that you want to include in your document. No actual images need to be put in your document itself as all the images reside outside of the XML document. So you should just include:
            1. Image-Label: Title of the image, e.g., “Figure 1: Input window for entering values of wing variables”.
            2. Filename: Name of file/URL of the image resource.

For example:

```
<image>
   <label> Figure 1: Input window for entering values of wing variables </label>
   <picture><filename>photo1.jpg</filename></picture>
</image>
```

The tags for the body part can be inserted as explained in step 6. The options corresponding to the body part are as shown below:

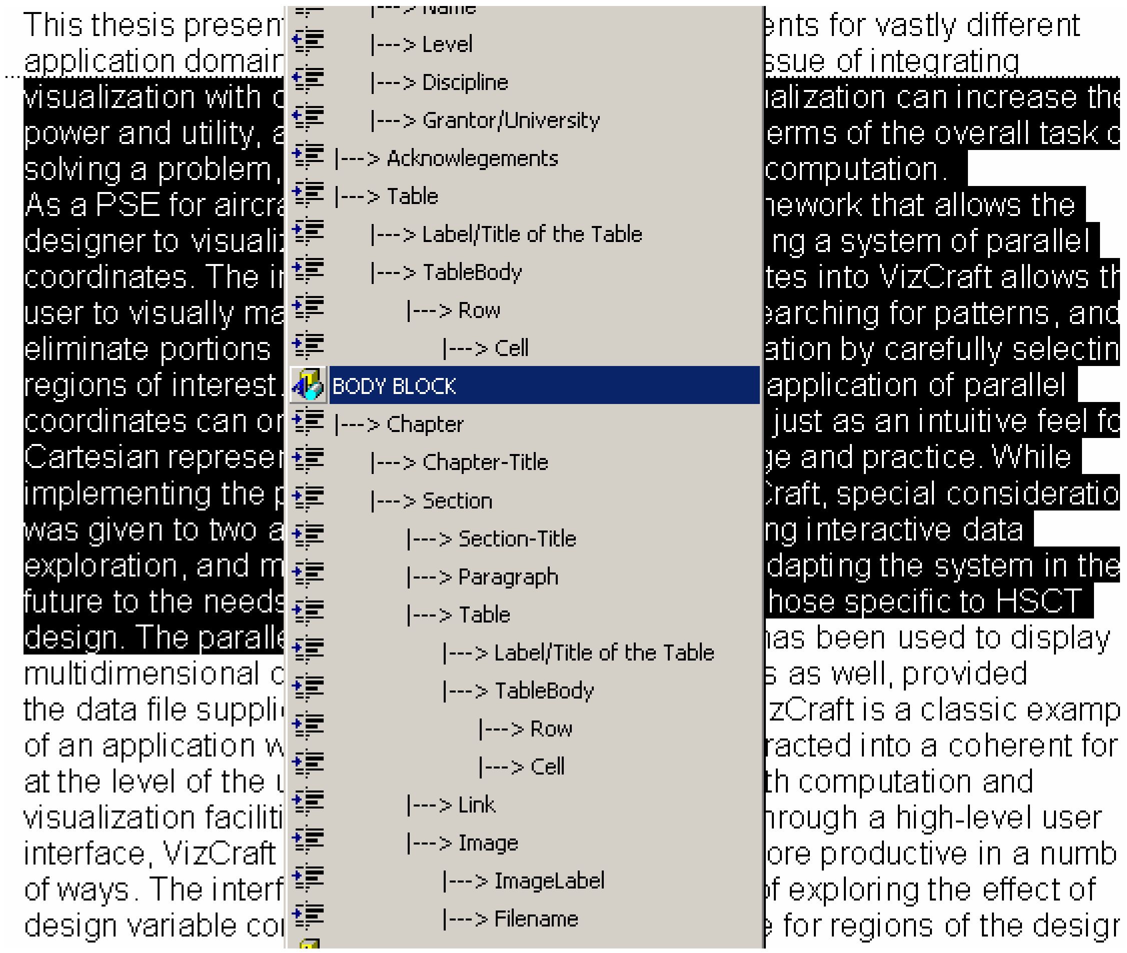


Figure 15: Macro tag options for the body portion

An example **body** portion of the thesis, after the tags have been inserted, will look like:
(Note that the original contents of the thesis have been modified for brevity)

<body>
<chapter>

<chapterTitle>Chapter 1</chapterTitle>

<section>
<sectionTitle>Introduction</sectionTitle>
<paragraph>
Scientists and engineers in many application domains commonly use modeling and simulation codes developed in-house that have poor documentation and a poor user interface. The code is often tied to a particular computing environment, and typically only the developers of the code can make effective use of it, reducing the productivity of many research groups. The recently proposed concept of problem solving environment (PSE) promises to provide
scientists and engineers with integrated environments for problem solving in the scientific domain, increasing their productivity by allowing them to focus on the problem at hand rather than on general computation issues.
</paragraph>
</section>

```xml
<section>
<sectionTitle>
1.1 What is a PSE?
</sectionTitle>

<paragraph>
A PSE is a system that provides a complete, usable, and integrated set of high level facilities for solving problems in a specific domain [34, 44]. PSEs allow users to define and modify problems, choose solution strategies, interact with and manage appropriate hardware and software resources, visualize and analyze results, and record and coordinate extended problem solving tasks. In complex problem domains, a PSE may provide intelligent and expert assistance in selecting solution strategies, e.g., algorithms, software components, hardware resources, data, etc. Perhaps most significantly, users communicate with a PSE in the language of the problem, not in the language of a particular operating system, programming language, or network protocol. Expert knowledge of the underlying hardware or software is not required. Experience in dealing with large-scale engineering design and analysis problems has indicated the critical need for PSEs with four distinguishing characteristics: (1) facilitate the integration of diverse codes, (2) support human collaboration, (3) support the transparent use of distributed resources, and (4) provide advisory support to the user. In principle, PSEs can solve simple or complex problems, support both rapid prototyping and detailed analysis, and can be used both in introductory education and at the frontiers of science.
</paragraph>
</section>
…

<section>

<sectionTitle>3.1 The HSCT Design Problem</sectionTitle>

<paragraph>
The design of the HSCT is an active research topic at the Multidisciplinary Analysis and Design (MAD) Center for Advanced Vehicles at Virginia Polytechnic Institute and State University. The design problem is to minimize the take-o_ gross weight (TOGW) for a 250 passenger HSCT with a range of 5; 500 nautical miles and a cruise speed of Mach 2:4. The simplified mission profile includes takeoff, supersonic cruise, and landing. For these efforts, a suite of low fidelity and medium fidelity analysis methods have been developed, which include several software packages obtained from NASA along with in-house software. A description of these tools is given by Dudley et al. [25]. We describe a PSE to aid aircraft designers during the conceptual design stage of the HSCT. Typically, the aircraft design process is comprised of three distinct phases: conceptual, preliminary, and detailed design. In the conceptual design stage, major design parameters for the final configuration are defined and set. The conceptual design phase models an aircraft with a set of values for significant parameters relating to the aircraft geometry, internal structure, systems, and mission. Examples of such parameters include the wing span, sweep, and thickness-to-chord ratios (t/c); the fuel and wing weights; the engine thrust; and the cruise altitude and climb rate, as shown in Table 3.1. Individual designs can be (and are) viewed as points in a multidimensional design space [86].
</paragraph>

<table>
<label>Table 3.1: Twenty-nine HSCT variables and their typical values</label>
<tablebody>
<row><cell>Number </cell><cell>Typical Value </cell><cell>Description</cell></row>
<row><cell>1 </cell><cell>181.48 </cell><cell>wing root chord, ft</cell></row>
<row><cell>2 <cell>155.9 </cell></cell><cell>leading edge break point, xft</cell></row>
<row><cell>3 </cell><cell>49.2 </cell><cell>leading edge break point, yft</cell></row>
```

```
<row><cell>4 </cell><cell>181.6 </cell><cell>trailing edge break point, xft</cell></row>
<row><cell>5 </cell><cell>64.2 </cell><cell>trailing edge break point, yft</cell></row>
<row><cell>6 </cell><cell>169.5 </cell><cell>leading edge wing tip, xft</cell></row>
<row><cell>7 </cell><cell>7.00 </cell><cell>wing tip chord, ft</cell></row>
<row><cell>8 </cell><cell>75.9 </cell><cell>wing semi-span, ft</cell></row>
</tablebody>
</table>

<paragraph>It is interesting to compare the aircraft design problem described here to other problems in multidimensional data analysis more frequently encountered. To illustrate this class of problems, consider locating a place to retire. There might typically be 10 or 20 variables to consider when evaluating possible retirement places, such as climate, population density, crime rate, etc. Data analysis for the retirement problem depends on building an objective function that attempts to assign values to each parameter on some linear scale and relative weights to the various parameters.
</paragraph>

<image>
<label>Figure 3.1: VizCraft design view window showing aircraft geometry and cross sections of the airfoil at the root, leading edge break, and tip of the wing. The panel on the left indicates the number of violated (red), active (yellow), and satisfied (green) constraints, and allows the user to modify values of design variables.</label>
<picture><filename>Fig3-1.jpg</filename></picture>
</image>

…

<paragraph>
I asked the students to complete a library-developed Web-based module on evaluating Internet resources
 <simpleLink href = "http://www.lib.vt.edu/research/libinst/idle/evaluating.html">
(http://www.lib.vt.edu/research/libinst/idle/evaluating.html)
 </simpleLink>
and to comment on anything new they saw in the module. The question I asked was, "Once you've looked at it, could you tell me if the module gives you information that you hadn't thought of - what ideas are new to you, what ideas are different, if it presents anything that you hadn't thought of before?" All of them found something in the module they hadn't previously considered.
</paragraph>

</section>
…

</chapter>
</body>
```

The **back** part would contain the end matter of an ETD. This typically includes:

1) Bibliography: This in turn consists of:
    a. Label/Title: e.g., "References"
    b. Reference: These are the actual references, e.g.:

       [98] P.C. Wong and R.D. Bergeron. Multiresolution multidimensional wavelet brushing. In Proceedings of IEEE Visualization '96, pages 141-148, New York, NY, October 1996. IEEE Computer Society Press.

2) Appendix: This in turn consists of:

a. Appendix Title: e.g., “Appendix A - Information Literacy Competency Standards”
b. Section: This is the same as the section in body and involves the main contents in form of :
   1. Section: A section can in turn have other sections recursively.
   2. Section-Title: This is the title of the section.
   3. Paragraph: These are the text portions of your content.
   4. Table: This in-turn consists of:
      1. Label/Title of the Table
      2. Table body: This in-turn consists one or more of:
         a. Row: which in-turn consists of:
            i. Cell: These are the different columns within a row.
   5. Link: This enables to add links in your document. Note that when you select a text and then tag it as Link a default form like this would appear (assuming the text you highlighted was “my-pic”):

      ```
      <simpleLink href=”filename”>my-pic</simpleLink>
      ```

      You then need to change the “filename” to the URL you want to refer to, e.g. “photo1.jpg”.

   6. Image: This is used in order to refer to images that you want to include in your document. No actual images need to be put in your document itself as all the images reside outside of the XML document. So you should just include:
      1. Image-Label: Title of the image, e.g., “Figure 1: Input window for entering values of wing variables “.
      2. Filename: Name of file/URL of the image resource.

For example:

```
<image>
   <label> Figure 1: Input window for entering values of wing variables</label>
   <picture><filename>photo1.jpg</filename></picture>
</image>
```

3) Vita: This in-turn consists of:
   a. Vita Title: e.g., “Vita”.
   b. Vita Author: e.g., “Aniket Prabhune”.
   c. Section: This is the same as the section in the body and involves the main contents in form of :
      1. Section: A section can in turn have other sections recursively.
      2. Section-Title: This is the title of the section.
      3. Paragraph: These are the text portions of your content.
      4. Table: This in-turn consists of :
         1. Label/Title of the Table
         2. Table body: This in-turn consists one or more of:
            b. Row: which in-turn consists of:
               i. Cell: These are the different columns within a row.
      5. Link: This enables to add links in your document. Note that when you select a text and then tag it as Link a default form like this would appear (assuming the text you highlighted was “my-pic”):

<simpleLink href=”filename”>my-pic</simpleLink>

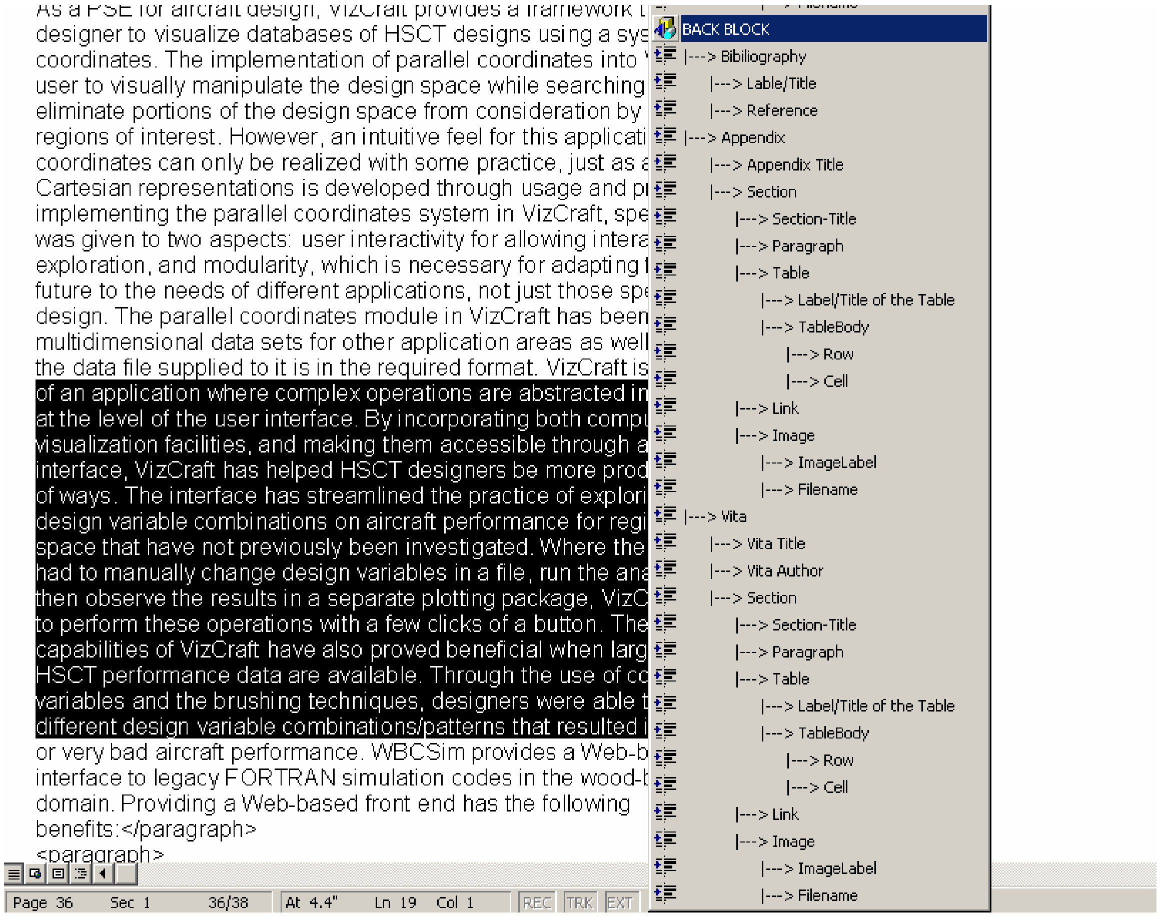
You then need to change the “filename” to the URL you want to refer to, e.g., “photo1.jpg”.

6. Image: This is used in order to refer to images that you want to include in your document. No actual images need to be put in your document itself as all the images reside outside of the XML document. So you should just include:
    1. Image-Label: Title of the image, e.g., “Figure 1: Input window for entering values of wing variables “
    2. Filename: Name of file/URL of the image resource

For example:

```
`          <image>
                <label> Figure 1: Input window for entering values of wing variables
                </label>
                <picture><filename>photo1.jpg</filename></picture>
           </image>
```

The tags for the **back** part can be inserted as explained in the step 6. The options shown corresponding to the **back** part are as shown below:



As a PSE for aircraft design, VizCraft provides a framework t
designer to visualize databases of HSCT designs using a sys
coordinates. The implementation of parallel coordinates into '
user to visually manipulate the design space while searching
eliminate portions of the design space from consideration by
regions of interest. However, an intuitive feel for this applicati
coordinates can only be realized with some practice, just as a
Cartesian representations is developed through usage and pr
implementing the parallel coordinates system in VizCraft, spe
was given to two aspects: user interactivity for allowing intera
exploration, and modularity, which is necessary for adapting t
future to the needs of different applications, not just those spe
design. The parallel coordinates module in VizCraft has been
multidimensional data sets for other application areas as well
the data file supplied to it is in the required format. VizCraft is
of an application where complex operations are abstracted in
at the level of the user interface. By incorporating both compu
visualization facilities, and making them accessible through a
interface, VizCraft has helped HSCT designers be more prod
of ways. The interface has streamlined the practice of explori
design variable combinations on aircraft performance for regi
space that have not previously been investigated. Where the
had to manually change design variables in a file, run the ana
then observe the results in a separate plotting package, VizC
to perform these operations with a few clicks of a button. The
capabilities of VizCraft have also proved beneficial when larg
HSCT performance data are available. Through the use of co
variables and the brushing techniques, designers were able t
different design variable combinations/patterns that resulted i
or very bad aircraft performance. WBCSim provides a Web-b
interface to legacy FORTRAN simulation codes in the wood-t
domain. Providing a Web-based front end has the following
benefits:</paragraph>
<paragraph>

Figure 16: Macro tag options for the back portion

An example **back** portion of the thesis, after the tags have been inserted, will look like:
(Note that the original contents of the thesis have been modified for brevity)

```
<back>

<bibliography>

<referenceTitle>Bibliography</referenceTitle>
<reference>
[1] R. Akers, E. Kanta, C.J. Randall, S. Steinberg, and R.L. Young. SciNapse: A problem solving
environment for partial differential equations. IEEE Computational Science and Engineering,
4(3):32-42, 1997.
</reference>
<reference>
[2] G. Allen, T. Goodale, and E. Seidel. The Cactus computational collaboratory: Enabling
technologies for relativistic astrophysics, and a toolkit for solving PDEs by communities in science
and engineering. In 7th Symposium on the Frontiers of Massively Parallel Computation, pages
36-41, Los Alamitos, CA, 1999. IEEE Computer Society Press.
</reference>

…

<reference>[100] J.V. Zweber, M. Blair, H. Kamhawi, G. Bharatram, and A. Hartong. Structural
and manufacturing analysis of a wing using the adaptive modeling language. In 39th
AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, pages
483-490, Long Beach, CA, AIAA-98-1758, April 1998.
</reference>
</bibliography>

<vita>
<vitaTitle>Vita</vitaTitle>

<vitaBody>On a winter evening, 9 December 1974, a baby boy was born to the Goel family in
Ghaziabad, UP, India. He was nameless for a week, until his parents finally decided on Amit. And
so his life story begins. He became interested in computers in the fifth grade, when computers
were first introduced in schools in India, and has remained passionate about computers ever
since. He received his Bachelor of Technology degree in Electrical Engineering from Indian
Institute of Technology Delhi in 1997, and proceeded to Virginia Tech for graduate study. He will
be graduating with a Master of Science degree in Computer Science in June 1999.
</vitaBody>

</vita>
</back>
```

8) After the entire document has been tagged press the update tags button. This will add the XML metadata tags at the top of the document and also an ending tag </thesis> at the end.

9) Now save the document as "Thesis.xml" (or use any filename you want, but you should have an .xml extension), selecting the file type as Text-Only.

10) The XML file has been created and recorded. Because some old browsers do not render the XML documents directly in the browser, you might want to convert it into HTML for rendering. For this use the Sablotron utility that you downloaded from the web. On the command prompt use the following command:

```
C:\thesis\ sabcmd <.xsl filename> <.xml filename> <.html filename>
```

For example:

```
C:\thesis\ sabcmd etd.xsl Thesis.xml Thesis.html
```

APPENDIX B: Reference theses created using this method

1) The XML File: http://www.ndltd.org/xml/etd/etd1/Thesis1.xml
   The HTML File: http://www.ndltd.org/xml/etd/etd1/Thesis1.html

2) The XML File: http://www.ndltd.org/xml/etd/etd2/Thesis2.xml
   The HTML File: http://www.ndltd.org/xml/etd/etd2/Thesis2.html

APPENDIX C: Virginia Tech DTD for ETDs

```
<!--
# Project: Electronic Thesis or Dissertation (ETD) in XML.   This document contains the DTD
used for ETDs according to the Template Approach by Tejas Patel. It was prepared to aid the
Graduate School, Virginia Polytechnic and State University. For more reference on the base
document used, please visit: http://www.ndltd.org/etd/etd-ml/dtdetds.htm
-->
<!--
************************************************************************
Element ETD is the root element of our DTD.
************************************************************************
-->
<!ELEMENT ETD (FRONT, MATTER, BACK)>
<!--
************************************************************************
ELEMENT  ETD is divided into 3 main elements. The front  matter (such as certificate of
approval, abstract, title pages,  epigraphs, etc.), which has primary content (chapters, which are
here  designated as "section"), and back (the bibliography and appendix). This permits the
assembly of the ETD in multiple files. Effectively, there will be one "intro.xml" and  several
"matter.html" files, and one "back.xml" file. So that same DTD can be used with each.
************************************************************************
-->
<!ELEMENT FRONT (TITLEPAGE,ACKNOWLEDGEMENTS?,ABSTRACT,TOC?,PREFACE?)>
<!-- This constitutes the first few pages of our ETD -->

<!ELEMENT ACKNOWLEDGEMENTS (#PCDATA)>

<!ELEMENT ABSTRACT (#PCDATA)>

<!-- Table of contents -->
<!ELEMENT TOC (ul)>

<!ELEMENT ul (li, (linebreak, li, indent,doubleindent)*)>
<!ELEMENT li ANY>
<!ELEMENT indent ANY>
<!ELEMENT doubleindent ANY>

<!ELEMENT PREFACE (#PCDATA)>
<!-- Preface of the Document -->

<!ELEMENT TITLEPAGE
(DOCTITLE|DOCAUTHOR+|WORKTITLE|SUBJECT|THESISADVISOR+|DATE|LOCATION|KEY
WORDS|THESISCOPYRIGHT)*>
<!-- this contains the details that appear on the first page of the ETD-->
<!ELEMENT DOCTITLE (#PCDATA)>
<!-- Represents the Title of the ETD-->

<!ELEMENT DOCAUTHOR (#PCDATA)>
<!-- There must be atleast one Document author and there might be more than one Document
Authors-->

<!ELEMENT WORKTITLE (#PCDATA)>
```

```
<!ELEMENT SUBJECT (#PCDATA)>
<!-- Represents the Subject of the ETD -->

<!ELEMENT THESISADVISOR (#PCDATA)>
<!-- Name of advisor for your thesis -->

<!ATTLIST THESISADVISOR role CDATA #REQUIRED>
<!-- What role did each thesis advisor play for this thesis-->

<!ELEMENT THESISCOPYRIGHT (#PCDATA)>
<!-- copyright information -->

<!ELEMENT DATE  (#PCDATA)>
<!-- The date the ETD was published-->

<!ELEMENT LOCATION (#PCDATA)>
<!--The location where Thesis was published -->

<!ELEMENT KEYWORDS (#PCDATA)>
<!-- Some key elements present in the Thesis and Dissertation -->

<!ELEMENT BACK (BIBLIOGRAPHY,APPENDIX*,CITATION?)>
<!-- Contains the Back matter of the ETD-->

<!ELEMENT BIBLIOGRAPHY (#PCDATA)>

<!ELEMENT APPENDIX (#PCDATA)>

<!ELEMENT CITATION (#PCDATA)>

<!ELEMENT MATTER (SECTION+,CHAPTERTITLE*)>
<!--  contains the main matter of the Thesis-->

<!ELEMENT SECTION (HEADING,(SUBSECTION | img | a | LIST)*)>
<!-- Matter is divided into sections with one heading and multiple subsections-->


<!ELEMENT CHAPTERTITLE (#PCDATA)>
<!ELEMENT HEADING (#PCDATA)>

<!ELEMENT SUBSECTION (HEADING,(SUBSECTION |img | a | LIST| linebreak)*)>
<!-- Each subsection in turn can contain multiple subsections and associated heading-->

<!ELEMENT img EMPTY>
<!ATTLIST img
        src CDATA #REQUIRED
        alt CDATA #IMPLIED
        width CDATA #IMPLIED
        height CDATA #IMPLIED
        border CDATA #IMPLIED
        align (left | right | top | bottom | middle) #IMPLIED
        vspace CDATA #IMPLIED
>

<!ELEMENT a (#PCDATA | img)*>
```

```
<!ATTLIST a
        href CDATA #IMPLIED
        name CDATA #IMPLIED
        target CDATA #IMPLIED
        lang CDATA #IMPLIED
>

<!ELEMENT LIST (ITEM+)>

<!ELEMENT ITEM (#PCDATA)>

<!-- Table Information -->
<!ELEMENT table (tr | th | td | caption)*>

<!ELEMENT td (#PCDATA | img | a | table )*>

<!ELEMENT tr (td | th | tr)*>

<!ELEMENT th (#PCDATA)>

<!-- End of DTD -->
```

APPENDIX D: Virginia Tech XSD (Schema) for ETDs

```
<!--

This document was originally created by:

Robert France, Hussein Suleman

Digital Library Research Laboratory, Virginia Polytechnic Institute and State University

{france,hussein}@vt.edu

It was then modified by the Students of CS5604, Fall 2001, Virginia Polytechnic Institute and
State University, taught by E.A.Fox:

Matthew Aguirre
Jiunwei Chen
J. Scott Masterson
David McPherson
David Tessendorf

It has been further modified by Aniket Prabhune, Graduate Student, Computer Science
Department, Virginia Polytechnic Institute and State University, Spring 2002.

-->

<?xml version="1.0" encoding="UTF-8" ?>

<schema xmlns="http://www.w3.org/2001/XMLSchema"
   xmlns:ETD_MS="http://www.ndltd.org/standards/metadata/etdms/1.0/"
   targetNamespace="http://www.ndltd.org/standards/metadata/etdms/1.0/"
 xmlns:xml="http://www.w3.org/XML/1998/namespace"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 elementFormDefault="qualified"
 attributeFormDefault="unqualified">

<import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.ndltd.org/standards/metadata/etdms/1.0/etd.xsd"/>
   <complexType name="simpleLink">
     <annotation>
       <documentation>
         This is similar to an anchor tag and allows to specify links within the ETD.
       </documentation>
     </annotation>
     <attribute name="xlink:type" type="xsd:string" use="fixed" value="simple" />
      <attribute ref="xlink:type" />
            <attribute ref="xlink:href" use="required" />
            <attribute ref="xlink:actuate" />
            <attribute ref="xlink:show" />
     </complexType>

    <complexType name="freeTextType">
```

```
  <annotation>
    <documentation>
            When a free text field is translated by someone other than the author, that person's
            name should appear as the value to the translated attribute.
    </documentation>
  </annotation>
  <simpleContent>
    <extension base="string">
      <attribute name="translated" type="string"/>
      <attributeGroup ref="xml:specialAttrs"/>
    </extension>
  </simpleContent>
</complexType>

<simpleType name="integer">
  <annotation>
    <documentation>
      As defined in ISO 8601 and the profile recommended for implementing ISO 8601 dates in
      Dublin Core.
    </documentation>
  </annotation>
  <restriction base="string">
    <pattern value="[0-9]+"/>
  </restriction>
</simpleType>

<complexType name="row">
    <annotation>
         <documentation>
            row - is an actual row in the table
            cell - is the actual entry in the table
         </documentation>
    </annotation>
    <sequence>
        <element name = "cell" type="ETD_MS:freeTextType" minOccurs="1"
         maxOccurs="unbounded" />
    </sequence>
  </complexType>

  <complexType name="tablebody">
       <annotation>
            <documentation>
               This represents the body of the table which consists of rows and cells.
            </documentation>
       </annotation>
       <sequence>
           <element name = "ETD_MS:row"  type="ETD_MS:freeTextType" minOccurs="1"
            maxOccurs="unbounded"/>
       </sequence>
  </complexType>

  <complexType name="table">
       <annotation>
            <documentation>
           label - is for something like "Table 1"
           tablebody - is the actual table
```

```
          <documentation>
        </annotation>
        <sequence>
            <element name = "label" type="ETD_MS:freeTextType" minOccurs="0"
              maxOccurs="1" />
            <element ref = "ETD_MS:tablebody" minOccurs="1" maxOccurs="unbounded" />
        </sequence>
       </complexType>

        <complexType name="picture">
        <annotation>
          <documentation>
             Includes the Name of the file/URL of the image source.
          </documentation>
        </annotation>
        <sequence>
            <element name="filename" type="ETD_MS:freeTextType" minOccurs="1"
              maxOccurs="1" />
        </sequence>
        </complexType>

        <complexType name="image">
        <annotation>
          <documentation>
              Used to refer to images that are to be included in the document.
          </documentation>
        </annotation>
         <sequence>
             <element name = "label" type="ETD_MS:freeTextType" minOccurs="0"
                   maxOccurs="1" />
             <element name = "picture"/>
         </sequence>
        </complexType>

<complexType name = "section" mixed="true" >
     <annotation>
          <documentation>
                  A section that can in turn have other sections recursively.
          </documentation>
     </annotation>
      <all>
          <element ref = "ETD_MS:section"    minOccurs="0" maxOccurs="unbounded" />
          <element name = "sectionTitle" type ="ETD_MS:freeTextType" minOccurs="0"
            maxOccurs="1" />
          <element name = "paragraph"  type="ETD_MS:freeTextType" minOccurs="0"
            maxOccurs="unbounded" />
          <element ref = "ETD_MS:table"      minOccurs="0" maxOccurs="unbounded" />
          <element ref = "ETD_MS:simpleLink" minOccurs="0" maxOccurs="unbounded"/>
          <element name = "ETD_MS:image"     minOccurs="0" maxOccurs="unbounded" />
     </all>
     </complexType>

   <complexType name="chapter">
        <annotation>
          <documentation>
             This represents a chapter in the thesis or dissertation and in turn consists of a chapter
          title and a number of sections.
        </documentation>
      </annotation>
       <sequence>
       <element name = "chapterTitle" type="ETD_MS:freeTextType" minOccurs="1"
         maxOccurs="1" />
       <element name = "section" type="ETD_MS:section"  minOccurs="1"
         maxOccurs="unbounded" />
       </sequence>
    </complexType>

    <complexType name="appendix">
      <annotation>
        <documentation>
          This represents the appendix in the back portion of the thesis or dissertation and
          includes an appendix title as well as sections as in the body of the thesis or
          dissertation.
        </documentation>
      </annotation>
      <sequence>
       <element name = "appendixTitle" type="ETD_MS:freeTextType" minOccurs="1"
         maxOccurs="1" />
       <element name = "section" type="ETD_MS:section" minOccurs="1"
         maxOccurs="unbounded" />
     </sequence>
    </complexType>

      <complexType name="vita">
      <annotation>
        <documentation>
           This represents the vita in the back portion of the thesis or dissertation and
            includes a vita title, a vita author as well as sections as in the body of the thesis or
            dissertation.
        </documentation>
      </annotation>
       <sequence>
        <element name = "vitaTitle" type="ETD_MS:freeTextType" minOccurs="1"
          maxOccurs="1" />
        <element name = "vitaAuthor" type="ETD_MS:freeTextType" minOccurs="1"
          maxOccurs="1" />
        <element name = "section" type="ETD_MS:section" minOccurs="1"
         maxOccurs="unbounded" />
       </sequence>
      </complexType>

<element name="thesis">
    <complexType>
      <sequence>
       <element name="front" minOccurs="0" maxOccurs="1">
       <complexType>
         <sequence>
           <element name="title" type="ETD_MS:freeTextType"
            minOccurs="1" maxOccurs="unbounded">
            <annotation>
              <documentation>
             A name given to the resource. In the case of theses and
```

```
   dissertations, this is the title of the work as it
   appears on the title page or equivalent.
    </documentation>
  </annotation>
</element>
<element name="creator" type="ETD_MS:authorityType" minOccurs="1"
 maxOccurs="unbounded">
  <annotation>
    <documentation>
      An entity primarily responsible for making the content of the resource. In the case of
      theses or dissertations, this field is appropriate for the author(s) of the work. Like other
      names and  institutions, this field should be entered in free text form as it appears on
      the title page or equivalent, with a link to an authority  record if available.
    </documentation>
  </annotation>
</element>

<element name="subject" type="ETD_MS:controlledTextType" minOccurs="1"
   maxOccurs="unbounded">
  <annotation>
    <documentation>
      The topic of the content of the resource. In the case of theses and dissertations,
       keywords or subjects listed on the title page can be entered as free text. The
      "scheme" qualifier should be used to indicate a controlled vocabulary. For more
       information visit: http://www.ndltd.org/standards/metadata/current.html#qualifier
    </documentation>
  </annotation>
</element>

<element name="description" minOccurs="0" maxOccurs="unbounded">
  <annotation>
    <documentation>
      An account of the content of the resource. In the case of theses and dissertations, this
      is the full text of the abstract unless otherwise qualified.
   </documentation>
  </annotation>
  <complexType>
    <simpleContent>
      <extension base="ETD_MS:freeTextType">
        <attribute name="role"
         type="ETD_MS:descriptionRoleType"/>
      </extension>
    </simpleContent>
  </complexType>
</element>

<element name="contributor" minOccurs="0" maxOccurs="unbounded">
  <annotation>
    <documentation>
      An entity responsible for making contributions to the content of the resource. Typical
      use would be for co-authors of parts of the work as well as advisors or committee
      members. Co-authors of the entire work would be more appropriate for the creator
       field.
    </documentation>
  </annotation>
  <complexType>
```

```
    <simpleContent>
      <extension base="ETD_MS:freeTextType">
        <attribute name="role" type="string"/>
        <attribute name="resource" type="anyURI"/>
      </extension>
    </simpleContent>
  </complexType>
</element>

<element name="date" type="ETD_MS:freeTextType" minOccurs="1" maxOccurs="1">
  <annotation>
    <documentation>
      A date associated with an event in the life cycle of the resource. In the case of theses
      and dissertations, this should be the date that appears on the title page or equivalent
     of the work. It should be recorded as defined in ISO 8601 and the profile recommended
     for implementing ISO 8601 dates in Dublin Core.
    </documentation>
  </annotation>
</element>

<element name="type" type="ETD_MS:freeTextType" minOccurs="1"
   maxOccurs="unbounded">
  <annotation>
    <documentation>
      The nature or genre of the content of the resource. This field is used to distinguish the
      resource from works in other genres and to identify the types of content included in the
      resource. The string "Electronic Thesis or Dissertation" is recommended as one of the
      repeatable values for this element. In addition, specify types of content using the
      standard vocabulary found at: http://dublincore.org/documents/dcmi-type-vocabulary/.
      Degree and Education Level are now handled by the thesis.degree field.
    </documentation>
  </annotation>
</element>

<element name="format" type="ETD_MS:freeTextType" minOccurs="0"
   maxOccurs="unbounded">
  <annotation>
    <documentation>
      The physical or digital manifestation of the resource. In the case of an electronic thesis
      or dissertation, this should contain a list of the electronic format(s) in which the work is
      stored and/or delivered. Use the standard MIME type whenever possible (for a
      list of "registered" MIME types, visit
      ftp://ftp.isi.edu/in-notes/iana/assignments/media-types/media-types
       ). List as "unknown" if no format information is available, omit if the work is not
      available in electronic form.
    </documentation>
  </annotation>
</element>

<element name="identifier" type="string" minOccurs="1" maxOccurs="unbounded">
  <annotation>
    <documentation>
      An unambiguous reference to the resource within a given context. This can and should
      be used to provide a URI where the work can be viewed or downloaded.
      Persistent URNs such as PURLs (http://purl.org/) or Handles (http://handle.net/) are
      recommended.
    </documentation>
  </annotation>
</element>

<element name="language" type="string" minOccurs="0" maxOccurs="unbounded">
  <annotation>
    <documentation>
      A language of the intellectual content of the resource. This should be the primary
      language the work is recorded in. Portions of the larger work that appear in other
      languages should use the language qualifier. See Global Qualifiers. Language names
      themselves should be recorded using ISO 639-2 (or RFC 1766). If the language is not
      specified, it is assumed to be English (en).
    </documentation>
  </annotation>
</element>

<element name="rights" type="ETD_MS:freeTextType" minOccurs="0"
   maxOccurs="unbounded">
  <annotation>
    <documentation>
      Information about rights held in and over the resource. Typically, this describes the
      conditions under which the work may be distributed, reproduced, etc., how these
      conditions may change over time, and whom to contact regarding the copyright of the
      work.
    </documentation>
  </annotation>
</element>

<element name="degree" minOccurs="0" maxOccurs="1">
  <annotation>
    <documentation>
      The degree associated with the work. For example, MS, PhD
    </documentation>
  </annotation>
  <complexType>
    <sequence>
      <element name="name" type="ETD_MS:freeTextType" minOccurs="0"
        maxOccurs="unbounded">
        <annotation>
          <documentation>
            Name of the degree associated with the work as it appears within the work
            (example: Masters in Operations Research).
          </documentation>
        </annotation>
      </element>
        <element name="level" type="string" minOccurs="0"
        maxOccurs="unbounded">
        <annotation>
          <documentation>
            Level of education associated with the document. Example: bachelors, masters,
            doctoral, postdoctoral, other.
          </documentation>
        </annotation>
      </element>

      <element name="discipline" type="ETD_MS:freeTextType" minOccurs="0"
```

```
                maxOccurs="unbounded">
               <annotation>
                 <documentation>
                   Area of study of the intellectual content of the document. Usually this will be a
                  department name or sometimes a program.
                 </documentation>
               </annotation>
             </element>

             <element name="grantor" type="ETD_MS:authorityType" minOccurs="0"
                maxOccurs="unbounded">
               <annotation>
                 <documentation>
                  Institution granting the degree associated with the work. Like other institution
                  names, this  field should be entered in free text form as it appears on the title page
                  or equivalent, with a link to an authority record where available.
                 </documentation>
               </annotation>
             </element>
           </sequence>
         </complexType>
       </element>

       <element name="acknowledgement" type="ETD_MS:freeTextType" minOccurs="0"
          maxOccurs="unbounded">
         <annotation>
           <documentation>
             This represents the acknowledgement statement in the front portion of the thesis or
             dissertation.
           </documentation>
         </annotation>
       </element>

       <element name = "table" type="ETD_MS:table" minOccurs="0"
         maxOccurs="unbounded" />
         <annotation>
           <documentation>
             This represents the table that can be inserted and in turn consists of the label of the
table
             and the rows and cells.
           </documentation>
         </annotation>
       </element>
      </sequence>
    </complexType>
  </element>

  <element name="body" minOccurs="0" maxOccurs="unbounded">
    <complexType>
      <sequence>
         <element name="chapter" type="ETD_MS:chapter" minOccurs="0"
           maxOccurs="unbounded" />
      </sequence>
    </complexType>
  </element>
```

```
<element name="back" minOccurs="0" maxOccurs="1">
    <annotation>
       <documentation>
          This represents the back portion of the ETD.
       </documentation>
    </annotation>

   <complexType mixed = "true">
        <sequence>
          <complexType name="bibliography">
            <annotation>
                <documentation>
                  This represents the references to be included in the back portion of the ETD and
                   in turn consists of the Label/Title of the bibliography and references.
                </documentation>
            </annotation>
            <sequence>
              <element name = "referenceTitle" type = "ETD_MS: freeTextType" minOccurs = "0"
                maxOccurs = "1" />
              <complexType>
                 <sequence>
                    <element name = "reference" type = "ETD_MS:freeTextType" minOccurs = "0"
                     maxOccurs = "unbounded" />
                 </sequence>
              </complexType>
             </sequence>
           </complexType>
          <element name="appendix" type="ETD_MS:appendix" minOccurs="0"
            maxOccurs="unbounded" />
          <element name="vita"    type ="ETD_MS:vita" minOccurs="0"
            maxOccurs="unbounded"/>
         </sequence>
        </complexType>
       </element>
  </sequence>
 </complexType>
</element>
</schema>
```

APPENDIX E: Virginia Tech XSL (eXtensible Style Sheet Language) file for ETDs

```
<!--

This document was originally created by:

Students of CS5604, Fall 2001 taught by E.A. Fox at Virginia Polytechnic Institute and State
University:

Matthew Aguirre
Jiunwei Chen
J. Scott Masterson
David McPherson
David Tessendorf

It has been further modified by Aniket Prabhune, Graduate Student, Computer Science
Department, Virginia Polytechnic Institute and State University.

-->

<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:n1="http://www.ndltd.org/standards/metadata/etdms/1.0/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xmlns:xlink="http://www.w3.org/1999/xlink" >

<xsl:output method="html" />

<!--Instructions for formatting the top level thesis tag of the ETD into HTML-->

<xsl:template match="n1:thesis">
    <xsl:apply-templates select="n1:front" />
    <xsl:apply-templates select="n1:body" />
    <xsl:apply-templates select="n1:back" />
</xsl:template>

<xsl:template match="/">
    <html>
          <head/>
             <body>
                   <!-- will match against the thesis tag defined above -->
                   <xsl:apply-templates />
             </body>
    </html>
</xsl:template>



<!--Instructions for formatting the front portion tags of the ETD into HTML-->

<xsl:template match="n1:front">
```

```
<!--Instructions for formatting the title tag in the front portion of the ETD into HTML-->

            <xsl:for-each select="n1:title">
                  <span style="font-size:x-large">
                          <xsl:apply-templates />
                          <br/>
                  </span>
                 <xsl:if test="position()!=last()">
                          <br/>
                  </xsl:if>
            </xsl:for-each>

           <br/>

<!--Instructions for formatting the creator tag in the front portion of the ETD into HTML-->

            <xsl:for-each select="n1:creator">
                  <xsl:if test="position()=1">
                       <span style="text-decoration:underline">Author:</span>
                       <xsl:text disable-output-escaping="yes">&lt;ul&gt;</xsl:text>
                  </xsl:if>
                  <li>
                       <xsl:apply-templates />
                  </li>
               <xsl:if test="position()=last()">
                     <xsl:text disable-output-escaping="yes">&lt;/ul&gt;</xsl:text>
               </xsl:if>
            </xsl:for-each>


<!--Instructions for formatting the subject tag in the front portion of the ETD into HTML-->

           <xsl:for-each select="n1:subject">
                <xsl:if test="position()=1">
                         <span style="text-decoration:underline">Subject:</span>
                         <xsl:text disable-output-escaping="yes">&lt;ul&gt;</xsl:text>
                </xsl:if>
                <li>
                         <xsl:apply-templates />
                </li>
                         <xsl:if test="position()=last()">
                                      <xsl:text disable-output-escaping="yes">&lt;/ul&gt;</xsl:text>
                         </xsl:if>
            </xsl:for-each>

<!--Instructions for formatting the description tag in the front portion of the ETD into HTML-->

            <xsl:for-each select="n1:description">
                    <xsl:if test="position()=1">
                           <span style="text-decoration:underline">Description:</span>
                           <br/>
                           <br/>
                    </xsl:if>
                    <div>
                          <xsl:apply-templates />
                    </div>
```

```
        <br/>
    </xsl:for-each>

<!--Instructions for formatting the contributor tag in the front portion of the ETD into HTML-->

    <xsl:for-each select="n1:contributor">
        <xsl:if test="position()=1">
            <span style="text-decoration:underline">Committee:</span>
            <xsl:text disable-output-escaping="yes">&lt;ul&gt;</xsl:text>
        </xsl:if>
        <li>
            <xsl:apply-templates />
        </li>
        <xsl:if test="position()=last()">
            <xsl:text disable-output-escaping="yes">&lt;/ul&gt;</xsl:text>
        </xsl:if>
    </xsl:for-each>

<!--Instructions for formatting the date tag in the front portion of the ETD into HTML-->

    <span style="text-decoration:underline">Date:</span><br/>
    <xsl:value-of select="n1:date"/>
    <br/>
    <br/>
<!--Instructions for formatting the type tag in the front portion of the ETD into HTML-->

    <xsl:for-each select="n1:type">
        <xsl:if test="position()=1">
            <span style="text-decoration:underline">Type:</span>
            <xsl:text disable-output-escaping="yes">&lt;ul&gt;</xsl:text>
        </xsl:if>
        <li>
            <xsl:apply-templates />
        </li>
        <xsl:if test="position()=last()">
            <xsl:text disable-output-escaping="yes">&lt;/ul&gt;</xsl:text>
        </xsl:if>
    </xsl:for-each>

<!--Instructions for formatting the format tag in the front portion of the ETD into HTML-->


    <xsl:for-each select="n1:format">
        <xsl:if test="position()=1">
            <span style="text-decoration:underline">Format:</span>
            <xsl:text disable-output-escaping="yes">&lt;ul&gt;</xsl:text>
        </xsl:if>
        <li>
            <xsl:apply-templates />
        </li>
        <xsl:if test="position()=last()">
            <xsl:text disable-output-escaping="yes">&lt;/ul&gt;</xsl:text>
        </xsl:if>
    </xsl:for-each>

<!--Instructions for formatting the identifier tag in the front portion of the ETD into HTML-->
```

```
          <xsl:for-each select="n1:identifier">
                 <xsl:if test="position()=1">
                      <span style="text-decoration:underline">Identifier:</span>
                      <xsl:text disable-output-escaping="yes">&lt;ul&gt;</xsl:text>
                 </xsl:if>
                 <li>
                     <xsl:apply-templates />
                 </li>
                <xsl:if test="position()=last()">
                     <xsl:text disable-output-escaping="yes">&lt;/ul&gt;</xsl:text>
                </xsl:if>
          </xsl:for-each>

<!--Instructions for formatting the language tag in the front portion of the ETD into HTML-->

          <xsl:for-each select="n1:language">
                <xsl:if test="position()=1">
                     <span style="text-decoration:underline">Language:</span>
                     <xsl:text disable-output-escaping="yes">&lt;ul&gt;</xsl:text>
                </xsl:if>
                <li>
                     <xsl:apply-templates />
                </li>
                     <xsl:if test="position()=last()">
                         <xsl:text disable-output-escaping="yes">&lt;/ul&gt;</xsl:text>
                     </xsl:if>
          </xsl:for-each>

<!--Instructions for formatting the rights tag in the front portion of the ETD into HTML-->


          <xsl:for-each select="n1:rights">
                     <xsl:if test="position()=1">
                         <span style="text-decoration:underline">Rights:</span>
                         <xsl:text disable-output-escaping="yes">&lt;ul&gt;</xsl:text>
                     </xsl:if>
                     <li>
                         <xsl:apply-templates />
                     </li>
                     <xsl:if test="position()=last()">
                         <xsl:text disable-output-escaping="yes">&lt;/ul&gt;</xsl:text>
                     </xsl:if>
          </xsl:for-each>

<!--Instructions for formatting the degree tag in the front portion of the ETD into HTML-->

          <xsl:for-each select="n1:degree">
                <xsl:if test="position()=1">
                     Degree:
                </xsl:if>
                <br/>
                <table border="1">

<!--Instructions for formatting the name tag within the degree tag in the front portion of the ETD
into HTML-->
```

```
                  <xsl:for-each select="n1:name">
                        <tr>
                          <td>Name</td>
                          <td>
                              <xsl:apply-templates />
                          </td>
                        </tr>
                 </xsl:for-each>
```

<!--Instructions for formatting the level tag within the degree tag in the front portion of the ETD into HTML-->

```
                <xsl:for-each select="n1:level">
                      <tr>
                          <td>Level</td>
                          <td>
                              <xsl:apply-templates />
                          </td>
                     </tr>
                </xsl:for-each>
```

<!--Instructions for formatting the discipline tag within the degree tag in the front portion of the ETD into HTML-->

```
             <xsl:for-each select="n1:discipline">
                    <tr>
                        <td>Discipline</td>
                        <td>
                              <xsl:apply-templates />
                        </td>
                  </tr>
              </xsl:for-each>
```

<!--Instructions for formatting the grantor tag within the degree tag in the front portion of the ETD into HTML-->

```
             <xsl:for-each select="n1:grantor">
                   <tr>
                        <td>Grantor</td>
                        <td>
                          <xsl:apply-templates />
                        </td>
                   </tr>
                </xsl:for-each>
             </table>
         </xsl:for-each>
         <br/>
```

<!--Instructions for formatting the acknowledgement tag in the front portion of the ETD into HTML-->

```
        <xsl:for-each select="n1:acknowledgement">
```

```
            <xsl:if test="position()=1">
                <span style="text-decoration:underline">Acknowledgement:</span>
                <xsl:text disable-output-escaping="yes">&lt;ul&gt;</xsl:text>
            </xsl:if>
            <li>
                <xsl:apply-templates />
            </li>
            <xsl:if test="position()=last()">
                <xsl:text disable-output-escaping="yes">&lt;/ul&gt;</xsl:text>
           </xsl:if>
      </xsl:for-each>


<!--Instructions for formatting the table tag in the front portion of the ETD into HTML-->

      <xsl:for-each select="n1:table">
           <xsl:apply-templates />
      </xsl:for-each>

</xsl:template>

<!--End of Instructions for formatting the front portion tags of the ETD into HTML-->

<!--Instructions for formatting the body portion tags of the ETD into HTML-->

<xsl:template match="n1:body" >
   <hr />

<!--Instructions for formatting the chapter tag in the body portion of the ETD into HTML-->

   <xsl:for-each select="n1:chapter">
        <font size="5">
            <xsl:apply-templates select="n1:chapterTitle" />
        </font>
        <br/>
        <br/>
        <xsl:for-each select="n1:section">
           <xsl:apply-templates />
        </xsl:for-each>
   </xsl:for-each>
</xsl:template>

<!--Instructions for formatting the chapterTitle tag within the chapter tag in the body portion of
the ETD into HTML-->

<xsl:template match="n1:chapterTitle" >
    <xsl:apply-templates />
</xsl:template>

<!--Instructions for formatting the section tag within the chapter tag in the body and back portion
of the ETD into HTML-->

<xsl:template match="n1:section" >
    <xsl:apply-templates />
</xsl:template >
```

```
<!--Instructions for formatting the sectionTitle tag within the section tag in the body and back
portion of the ETD into HTML-->

    <xsl:template match="n1:sectionTitle" >
          <p>
            <font size="4">
               <xsl:apply-templates />
            </font>
          </p>
    </xsl:template >

<!--Instructions for formatting the paragraph tag within the section tag in the body and back
portion of the ETD into HTML-->

   <xsl:template match="n1:paragraph" >
          <p>
             <xsl:apply-templates />
          </p>
    </xsl:template>

<!--Instructions for formatting the table tag within the section tag in the body and back portion of
the ETD into HTML-->

    <xsl:template match="n1:table" >
          <xsl:apply-templates />
    </xsl:template >

<!--Instructions for formatting the tablebody tag within the table tag in the body and back portion
of the ETD into HTML-->

   <xsl:template match="n1:tablebody">
          <table border="1">
             <xsl:apply-templates />
          </table>
    </xsl:template>

<!--Instructions for formatting the row tag within the table tag in the body and back portion of the
ETD into HTML-->

    <xsl:template match="n1:row">
          <tr>
              <xsl:apply-templates />
          </tr>
    </xsl:template>

<!--Instructions for formatting the cell tag within the table tag in the body and back portion of the
ETD into HTML-->

    <xsl:template match="n1:cell">
          <td>
             <xsl:apply-templates />
          </td>
    </xsl:template>
```

```
<!--Instructions for formatting the simpleLink tag within the section tag in the body and back
portion of the ETD into HTML-->

    <xsl:template match="n1:simpleLink">
          <a>
            <xsl:attribute name="href">
                <xsl:value-of select="@href"/>
            </xsl:attribute>
            <xsl:value-of select="."/>
          </a>
   </xsl:template>


<!--Instructions for formatting the image tag within the section tag in the body and back portion
of the ETD into HTML-->

   <xsl:template match="n1:image">
       <xsl:apply-templates />
       <xsl:text disable-output-escaping="yes">&lt;img </xsl:text>
       <xsl:for-each select="n1:width">
          <xsl:if test="position()=1">
              width="<xsl:value-of select="." />"
          </xsl:if>
       </xsl:for-each>
       <xsl:for-each select="n1:height">
           <xsl:if test="position()=1">
              height="<xsl:value-of select="." />"
           </xsl:if>
       </xsl:for-each>
           <xsl:text disable-output-escaping="yes">src=&quot;</xsl:text>
           <xsl:value-of select="n1:picture" />
          <xsl:text disable-output-escaping="yes">&quot;&gt;</xsl:text>
   </xsl:template>

<!--Instructions for formatting the label tag within the image tag in the body and back portion of
the ETD into HTML-->

   <xsl:template match="n1:label">
      <p>
         <b>
            <xsl:apply-templates />
          </b>
       </p>
   </xsl:template>


<!--Instructions for formatting the picture tag within the image tag in the body and back portion
of the ETD into HTML-->

   <xsl:template match="n1:picture">
   </xsl:template>

<!--End of Instructions for formatting the body portion tags of the ETD into HTML-->
```

```
<!--Instructions for formatting the back portion tags of the ETD into HTML-->

   <xsl:template match="n1:back" >
       <xsl:apply-templates/>
       <br/>
   </xsl:template>

<!--Instructions for formatting the bibliography tag in the back portion of the ETD into HTML-->

   <xsl:template match="n1:bibliography">
        <xsl:apply-templates />
   </xsl:template>

<!--Instructions for formatting the referenceTitle tag within the bibliography tag in the back
portion of the ETD into HTML-->

    <xsl:template match="n1:referenceTitle">
       <hr/>
          <p>
             <h2>
                 <b>
                    <xsl:apply-templates />
                 </b>
             </h2>
          </p>
  </xsl:template>

<!--Instructions for formatting the reference tag within the bibliography tag in the back portion of
the ETD into HTML-->

    <xsl:template match="n1:reference">
       <p>
             <xsl:apply-templates />
       </p>
    </xsl:template>

<!--Instructions for formatting the appendix tag within the bibliography tag in the back portion of
the ETD into HTML-->

    <xsl:template match="n1:appendix">
        <xsl:apply-templates />
    </xsl:template>

<!--Instructions for formatting the appendixTitle tag within the appendix tag in the back portion of
the ETD into HTML-->

    <xsl:template match="n1:appendixTitle">
       <hr/>
           <p>
               <h2>
                   <b>
                       <xsl:apply-templates />
                   </b>
               </h2>
          </p>
  </xsl:template>
<!--Instructions for formatting the vita tag within the bibliography tag in the back portion of the
ETD into HTML-->

   <xsl:template match="n1:vita">
       <xsl:apply-templates />
   </xsl:template>

<!--Instructions for formatting the vitaTitle tag within the vita tag in the back portion of the ETD
into HTML-->

    <xsl:template match="n1:vitaTitle">
       <hr/>
          <p>
             <h2>
                <b>
                      <xsl:apply-templates />
                </b>
             </h2>
           </p>
   </xsl:template>

<!--Instructions for formatting the vitaAuthor tag within the vita tag in the back portion of the ETD
into HTML-->

   <xsl:template match="n1:vitaAuthor">
       <p>
          <h3>
              <b>
                <xsl:apply-templates />
              </b>
          </h3>
        </p>
   </xsl:template>

</xsl:stylesheet>
```

References

1) Fox, E. A., Networked Digital Library of Theses and Dissertations, www.ndltd.org, An initiative for improving graduate education by developing accessible digital libraries of theses and dissertations

2) Moxley Joseph, www.etdguide.org, A resource for graduate students who are writing theses or dissertations, for graduate faculty who want to mentor ETD authors, for graduate deans who want to initiate ETD programs, and for IT administrators at universities.

3) Fox, E.A., McMillan Gail, Electronic Thesis and Dissertation initiative at Virginia Tech, http://etd.vt.edu/, A website explaining ETD preparation and submission process at Virginia Tech.

4) Independent Study Report of Ms. Dilshad Akhter (http://www.ndltd.org/xml/Dilshad/XML_ETD_Home.htm)

5) Independent Study Report of Mr. Tejas Patel on XML for ETDs, Spring 2001 (http://www.ndltd.org/xml/Tejas/Report.doc)

6) Report of the XML-ETD group in CS5604, Fall 2001, supervised by E. Fox (http://rocky.dlib.vt.edu/~cs5604/fall_2001/etd_xml/)

7) Information about XML : http://www.w3.org/XML/

8) XML 1.0 Recommendation: www.w3.org/TR/REC-xml

9) Specific articles on XML : www.xml.com

10) Tittel, E., Boumphrey, F., XML for Dummies, IDG,2000

11) Phillips, L.,A., "Using XML", QUE, 2000

12) Ceponkus A., Hoodbhoy, F., "Applied XML", Wiley, 1999

13) XML Tutorials:

Topics	**Related Websites**
1) Basic Concepts of XML	http://archive.devx.com/projectcool/developer/xmlz/ http://wdvl.com/Authoring/Languages/XML/
2) XSL, XSLT, XLink, XPointer, XML Schema	http://wdvl.com/Authoring/Languages/XML/
3) XML Mailing Lists	www.w3.org/XML/#discussion www.oasis-open.org/cover/lists.html#discussionLists