

An Introduction to XML **eXtensible Markup Language**

Advance Database Concepts

ADVANDB – CIS559M

In Partial Fulfillment for the Degree of MSIT
Master of Science in Information Technology
De La Salle University Manila
College of Computer Science

Submitted by:

Jinno Ordonez

Submitted to:

Dr. Remedios Bulos

August 9, 2005

TABLE of CONTENTS

I. History and Evolution of XML	
a. XML and HTML	1
b. The History of Markup	2
c. Procedural Markup to Generic Coding	3
d. SGML, DTD and HTML	3
e. Structuring XML	4
II. Description of XML & Related Issues	6
a. eXtensible Markup Language	
i. XML Document Types	
ii. Creating XML Documents: Elements, Attributes and Values	
iii. XML Five Commandments	
iv. Special XML Symbols	
v. The XML Declaration	
b. Defining XML Data	7
i. XML Schemas and Data Modelling	
1. DTDs and XSDs	
ii. Comparing DTDs and XSDs	
iii. XML Namespaces	
c. Formatting and Displaying XML Documents	14
i. CSS, XSLT and XSLFO	
ii. XSLT Sample	
d. Validating Documents	16
i. Well-Formed or Valid	
e. Processing and Managing XML Data	17
i. Parsers	
ii. DOM and SAX	
III. Current State of XML & Review of Related Works	18
a. WML (Wireless Markup Language)	19
b. XML Protocols	19
c. Using XML with Database	
IV. Applications of XML	
a. Microsoft and XML	20
b. IBM: The Lotus Knowledge Discovery Systems	21
V. Future Trends and Research Issues of XML	22
a. XHTML	22
b. Microsoft and Closed Source	

References

I. HISTORY AND EVOLUTION OF XML

A. XML and HTML

XML was developed by W3C¹ (World Wide Web Consortium) not to replace HTML but to further improve the capabilities of HTML. HTML has become the main markup language used in the web today, with over 800 million pages based on HTML. HTML has become so huge that its latest version has close to 100 tags. Why has HTML grown so big? The web began as a means to publish scientific documents. Today, it has grown to become a full-fledged medium comparable to TV and print ads. But what makes the web uniquely different from TV and print ads is the fact that it is interactive. HTML began popular because of its feature called the 'Hyperlink'². This feature was later further developed to support 'Hypermedia'³.

In support to these features HTML has also risen to support new technologies that were introduced to the advancement of web. Such technologies are JavaScript, Java, Flash, Server-Side Scripting (CGI, ASP, JSP, PHP), streaming media (WMV, AVI, MPEG), audio formats (MP3, WMA, WAV) and so much more.

So far I have discussed the developments in terms of software program mediums. But how will this change if we the medium itself (hardware) was altered? "W3C expects that by the year 2002, 75% of surfers won't be using the PC (Marchal, 2000)." What's worth noting in that quote is the fact that people will not be constraint to surf the web using the PC. Such technologies as PDAs, cellular phones, portable audio devices (Apple iPod, iRiver, Creative) and even gaming consoles (Sony Playstation 3, Microsoft Xbox, Nintendo Game Cube) will make use of the web. The problem with some of these devices, especially PDAs and cellular phones is that they are not as powerful as the PCs therefore they have difficulties processing HTML. Another problematic situation is that it is very common to find pages with more tags than the actual content of the page. This will take longer time to view the page in comparison to pages with less tags.

It is clear at this point that HTML as a markup language has become such a complex language and the question is should the language be further enhance and improved or is it time to seek alternatives to the language? The answer of the W3C is that, it is time to seek alternatives, hence XML was born.

¹ W3C is the organization in charge of the development of Web standards, mostly HTML (Marchal, 2000).

² Hyperlink is a feature that allows users to jump/navigate from one page to another.

³ Hypermedia is a feature that allows users to open different types of medium/formats such as animation, video stream, text, and highly graphical pages.

It must be clearly understood that XML was not developed to replace HTML. As described in the previous paragraphs, we can say that the technology was developed to support the areas in which HTML cannot further address. It can be said that XML was developed because HTML was such a success. "XML is therefore unlikely to replace HTML (Marchal, 2002)".

Some areas where XML is useful:

- Web site maintenance. XML works in the backend simplifying the HTML design.
- Information exchange between organizations
- Offloading and reloading of databases
- Syndicated content, where content is being made available to different Web sites
- Electronic commerce applications where different organizations collaborate to serve a customer
- Scientific applications with new markup languages for mathematical and chemical formulas
- Electronic books with new markup languages to express rights and ownership
- Handheld devices and smart phones with new markup languages optimized for these "alternative" devices.

(Marchal, 2002)

B. THE HISTORY OF MARKUP

Before we proceed into the content, let's first define SGML, HTML and XML.

1. SGML – Standard Generalized Markup Language, a specification for creating markup languages in a standard way.
2. HTML – a markup language constructed according to the SML standard. Thus, it is an SGML application.
3. XML – is a subset of SGML that is particularly suited to creating markup languages for the Internet. Thus, it is not a markup language but a specification for creating markup languages for the Internet.

(Mercer, 2001)

Now what is a markup language? It was earlier discussed that the web was developed primarily to be able to publish scientific documents and HTML was developed because of this. What's the reason? Scientific documents just like any formal documents requires standards in which it is to be created. Such document standards are titles, headings, reference, footnotes, etc. Some even have templates such as where the image should be displayed, how the text should look (italicized, bold, size 12 font, Times New Roman font, etc.) and where should it be located in the page. This are all the reasons why a markup language was developed, therefore a markup language is a way for developers to format/display the content in a web page in

accordance to the user. Mark-up was introduced in the publishing industry. The manuscript would provide layout instructions for the typesetter. The manuscript would be handwritten with the included instructions. These annotations are called mark-up⁴. This usually happens before and after the typesetter typed the document.

Procedural Markup to Generic Coding

When we use such word processing programs such as Microsoft Word it is easy to define how we format text. We simply click such features as the 'Bold' button, 'Italicized' button, format paragraph to 1.5 lines, indent, etc. These simple commands automatically generate the markup so the user does not have to code them. This is simply known as procedural markup. This later evolved to a more complex structure called Generic Coding. Generic coding allows the user to specify the code used for creating formatted templates. This comes in a form called Macros. Microsoft Word 2003 Help would describe Macros and its use as:

- If you perform a task repeatedly in Microsoft Word, you can automate the task by using a macro. A macro is a series of Word commands and instructions that you group together as a single command to accomplish a task automatically.
- Here are some typical uses for macros:
 - o To speed up routine editing and formatting
 - o To combine multiple commands; for example, inserting a table with a specific size and borders, and with a specific number of rows and columns
 - o To make an option in a dialog box more accessible
 - o To automate a complex series of tasks
- Word offers two ways to create a macro: the macro recorder and the Visual Basic Editor.

SGML, DTD and HTML

SGML (Standard Generalized Markup Language) was developed by Dr. Charles Goldfarb from IBM. SGML is published as an international standard by ISO. SGML further extends the capabilities of generic coding. SGML can be said to be similar to generic coding but with 2 additional characteristics:

- the markup describes the document's structure, not the document appearance
- the markup conforms to a model, which is similar to a database schema. This means that it can be processed by software or stored in a database.

(Marchal, 2002)

SGML does not create a standard that every document can follow. That's just an impossible task to create a single document in which all document will follow its structure. The approach of SGML is to allow the authors to create their own structure. SGML is a language used to

⁴ "Electronic markup is spelled as one word to distinguish it from traditional handwritten mark-up (Marchal, 2002)."

describe documents. It is almost similar to program languages, so it's actually very flexible and open to new applications (Marchal, 2002).

There are 2 notable applications created from SGML. These are the famous HTML and DTD or better known as SGML application. I will discuss DTD in detail in next section of this paper.

STRUCTURING XML

It is known that XML is a very powerful application program though there is a reason why it is still not as widely accepted as it should be. It still strikes as scary to most developers and those who do adopt it have been poorly educated on its structure. To better make sense of its structure we can divide the XML technologies into 3 layers:

1. XML Grammars
2. XML Protocols
3. XML Standards

XML can then be further subdivided into:

1. XML Standards

There are currently 6 components of the XML standards family, which comprise the following:

- a. XML Specification
- b. Extensible Stylesheet Language Transformation (XSLT)
- c. XML Path (XPath)
- d. Document Type Definition (DTD)
- e. XML Namespace
- f. XML Schema

2. XML Protocols

The XML protocols define how protocols can be designed and built using XML technologies.

Some of the XML protocols include:

- a. Simple Object Access Protocol (SOAP):
- b. XML RPC: A Remote Procedure Calling Protocol
- c. ebXML: ebXML is a set of specifications that together enable a modular electronic business framework through the exchange of XML messages.
- d. BizTalk: The Microsoft BizTalk Framework is an XML framework for application integration and electronic commerce. It is an industry initiative started by Microsoft and support by a wide range of organizations, from technology vendors like SAP and CommerceOne to technology users like Boeing and BP/Amoco.

3. XML Grammars

The XML grammars define the use of XML for specific applications. Some of the grammars of XML are as follows:

- a. Wireless Markup Language (WML)
- b. Extensible HTML (XHTML)
- c. Voice Extensible Markup Language (VoiceXML)

The Division of XML (Foo & Lee, 2002)

II. DESCRIPTION OF THE XML AND RELATED ISSUES

A. eXtensible MARKUP LANGUAGE

"The oldest working draft regarding XML currently listed at the W3C , dated November 14, 1996. It says XML is to be an extremely simple version of SGML that will allow generic SGML to be served, received, and processed on the Web like HTML (Mercer, 2001). "

It is important to make clear that XML is a subset of SGML. It basically holds the same concept as SGML, in which it takes into consideration the structure of documents and applications. HTML on the other hand is an application of SGML as well as DTD which is sometimes called SGML applications as mentioned earlier in this paper.

XML Document Types

There are 2 classification of XML types of document:

- 1. Document Applications** - these applications allow for human to manipulate the information. These are your usual document publishing. The main objective of XML in this classification is the build a rigid document structure to avoid human error.
- 2. Data Applications** - these applications are intended for software manipulation. This takes into consideration integrating together the "application as a document (Marchal, 2000)" seamlessly. As we have seen a structure of a document can be easily transformed into a table in a database by applying XSLT.

Creating XML Documents: Elements, Attributes and Values

There are 3 fundamentals in the building blocks of XML: elements, attributes, and values. An element contains two tags an opening tag and a closing tag. Consider the following examples:

1. Opening tag: <pet>
2. Closing tag: </pet>

Elements are used as the beginning of creating your XML documents. They are used to describe the information you mean to display or contain.

You can also create what is known as empty elements:

3. Empty element: <friend/>

An empty element doesn't contain any specific information, though later on you can fill it up. Ones that contain information are nonempty elements which are in short elements. An empty element is usually used when it is not part of the element being described. For example you would not consider a 'shirt' to be part of the element 'apartment'.

Attributes are what is used to describe elements. In this example the attributes Pages and Type are describing the element Book.

4. `<Book Pages="357" Type="SOFTCOVER">`

Attributes can also be used to identify empty elements:

5. `<friend name="Augustus"/>`

The last fundamental is value. Value is what describes the attribute. In the sample, `<friend name="Augustus"/>`, the value is "Augustus".

XML's Five Commandments:

1. Tag names are case sensitive.
2. Every opening tag must have a corresponding closing tag (unless it is an empty tag).
3. A nested tag pair cannot overlap another tag.
4. Attribute values must appear within quotes.
5. Every document must have a root element.

(Morrison, 2002)

Special XML Symbols

There are special XML symbols that cannot simply be typed into the content if you want to use them. XML has what is called entities. These entities are created with '&' in the beginning and ';' at the end. There are five entities defined in the XML language:

1. `<` - `<`
2. `>` - `>`
3. `"` - `"`
4. `'` - `'`
5. `&` - `&`

The XML Declaration

Before beginning any XML document you must declare the document as an XML page by using:

```
<?xml version="1.0"?>
```

B. DEFINING XML DATA

The defining factor of XML is its structure and rigidity, which solves a lot of human error if not all in using XML. There is very little room to make errors in XML because of its simplicity but nevertheless, mistakes can be made therefore errors must be detected. Schema is what allows errors to be detected in XML, which allows developers to define the format and structure of XML data.

XML Schemas & Data Modelling

Schemas basically constraints the user in what elements and attributes they can use. A schema in general defines the XML markup language. Schemas allows developers to make use of XML in what its purpose really is: to create markup languages. Schemas therefore sets its own rules for the markup language, along with which what elements and attributes are available to the user. The process of creating schemas for XML documents is known as data modeling.

There are 2 main XML schema approaches which you can choose from:

1. DTDs (Document Type Definitions)
2. XSDs (XML Schemas)

DTDs and XSDs

DTDs as we know comes from the predecessor of XML which was SGML. DTDs is known as an SGML application. Consider the following example of a DTD for the Tall Tales XML Document:

```
<!ELEMENT talltales (tt) +>
<!ELEMENT tt (question, a,b,c)>
<!ATTLIST tt
    Answer (a | b | c)> ---- attributes of tt (only attribute)
<!ELEMENT question (#PCDATA)>
<!ELEMENT a (#PCDATA)>
<!ELEMENT b (#PCDATA)>
<!ELEMENT c (#PCDATA)>
```

Sample taken from (Morrison, 2002)

We can describe DTDs as something that is very cryptic, that's why W3C decided to create a simpler schema creator called XSD. XSD are more easily to understand because you create them using the same structure as creating XML documents. XSD is even more powerful in that it allows you to associate data types with elements such as integer numbers and dates. DTDs is limited to this feature. But of course, what comes with a more flexible and powerful orogram with advanced features is of course complexity. As compared to DTDs, XSDs aren't nearly as compact as DTDs.

An example of an XSD Document That Serves as a Schema for the Tall Text XML Document:

```
<?xml version="1.0"?>
<xsd: schema xmlns:xsd=http://www.w3.org/2000/10/XMLSchema>

<xsd: element name="talltales" minOccurs="1" maxOccurs="1">
<xsd :complexType>
  <xsd :element name= "tt">
    <xsd:sequence>
      <xsd:element name="question" type="xsd:string" maxOccurs="1"/>
      <xsd:element name="a" type="xsd:string" maxOccurs="1"/>
      <xsd:element name="b" type="xsd:string" maxOccurs="1"/>
      <xsd:element name="c" type="xsd:string" maxOccurs="1"/>
    </xsd:sequence>
    <xsd:attribute name="answer" type="answerType" use="required"/>
    <xsd:simpleType name="answerType">
    <xsd:restriction base="xsd:NMTOKEN">
      <xsd:enumeration value="a"/>
      <xsd:enumeration value="b"/>
      <xsd:enumeration value="c"/>
    </xsd:restriction>
    </xsd:simpleType>
  </xsd:complexType>
</xsd:element>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

Another sample with XML Output:

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema id="XMLSchema1" targetNamespace="http://tempuri.org/XMLSchema1.xsd"
elementFormDefault="qualified"
  xmlns="http://tempuri.org/XMLSchema1.xsd"
  xmlns:mstns="http://tempuri.org/XMLSchema1.xsd"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"></xs:schema>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" />
      <xsd:element name="title" type="string" />
      <xsd:element name="occupation" type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="birthdate" type="xsd:date" />
    <xsd:attribute name="weight" type="xsd:integer" />
  </xsd:complexType>
</xsd:element>

</xsd:schema>
```

XML Output:

```
<person birthdate="1969-10-28" weight="160">
  <name>Milton James </name>
  <title>Mr.</title>
  <occupation>mayor, chef </occupation>
</person>
```

All Samples taken from (Morrison, 2002)

A simpler example:

SCREEN/XML OUTPUT

```
<?xml version="1.0" encoding="UTF-8" ?>
- <book isbn="0836217462">
  <title>Being a Dog Is a Full-Time Job</title>
  <author>Charles M. Schulz</author>
  - <character>
    <name>Snoopy</name>
    <friend-of>Peppermint Patty</friend-of>
    <since>1950-10-04</since>
    <qualification>extroverted beagle</qualification>
  </character>
  - <character>
    <name>Peppermint Patty</name>
    <since>1966-08-22</since>
    <qualification>bold, brash and tomboyish</qualification>
  </character>
</book>
```

XSD CODE

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <!-- definition of simple types -->
  <xs:simpleType name="nameType">
    <xs:restriction base="xs:string">
      <xs:maxLength value="32"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="sinceType">
    <xs:restriction base="xs:date"/>
  </xs:simpleType>

  <xs:simpleType name="descType">
    <xs:restriction base="xs:string"/>
  </xs:simpleType>

  <xs:simpleType name="isbnType">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9]{10}"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- definition of complex types -->
  <xs:complexType name="characterType">
    <xs:sequence>
      <xs:element name="name" type="nameType"/>
      <xs:element name="friend-of" type="nameType" minOccurs="0"
maxOccurs="unbounded"/>
      <xs:element name="since" type="sinceType"/>
      <xs:element name="qualification" type="descType"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="bookType">
    <xs:sequence>
      <xs:element name="title" type="nameType"/>
      <xs:element name="author" type="nameType"/>
      <xs:element name="character" type="characterType" minOccurs="0"/>
      <!-- the definition of the "character" element is using the "characterType"
complex type -->
    </xs:sequence>

    <xs:attribute name="isbn" type="isbnType" use="required"/>
  </xs:complexType>

  <!-- Reference to "bookType" to define the "book" element -->
  <xs:element name="book" type="bookType"/>
</xs:schema>
```

Samples taken from (Vlist, 2001)

The elements of the XSD is composed of the following:

1. Header

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema id="XMLSchema1" targetNamespace="http://tempuri.org/XMLSchema1.xsd"
elementFormDefault="qualified"
xmlns="http://tempuri.org/XMLSchema1.xsd"
xmlns:mstns="http://tempuri.org/XMLSchema1.xsd"
xmlns:xs="http://www.w3.org/2001/XMLSchema"></xs:schema>
```

The heading contains the declaration of the XML document. The schema element (<xs:schema id=...>) opens our schema. This can also hold Namespaces and other options.

2. Complex Type & Compositors

There are two types of elements defined in XSD, the first one is **Complex Type**. The complex type is defined by elements that contain attributes and values. Complex Types are then followed by **compositors**. In this case, the compositor is 'sequence' which defines an ordered sequence of sub-elements. Other compositors are 'choice' and 'all'.

3. Simple Type

Simple Types are elements which do not have further attributes. Some options we can create with Simple Types are **restrictions** and **patterns**. Restrictions allow us to put a constraint on our data types. For instance we can restrict our data type string with a maximum of 32 characters. Patterns on the other hand defines a regular expression that must be matched. So if we have

(Vlist, 2001)

Comparing DTDs and XSDs

1. DTDs are coded using a special language, whereas XSDs are coded in XML.
2. XSDs can be processed just like any other XML documents.
3. XSDs support a variety of data types (integer, floating points, Booleans, dates and so on), whereas DTDs treat all data as strings or lists of strings.
4. XSDs present an open-ended data model, which allows you to extend custom markup languages and establish complex relationships between elements without invalidating documents. DTDs employ a closed data model that doesn't allow for much in the way of extensibility.

(Morrison, 2002)

XML Namespaces

The problem of XML is that it is too extensible. Allowing anyone to be able to create tags the problem of ambiguity rises (Author Unknown, 'XML References'). For instance if you have two tags named <title>, the element can be referred to either as 'title of the book' or 'title of a person'. Namespace does not limit the user to create only one instance of the element <title>, but instead it allows you create more but it needs to be unique. The solution for example by the use of <title> we can create two different tags as:

- <qa: title> The Book Name </qa: title>
- <pa: title> Mr. </pa: title>

The solution is by defining there prefixes as references, they must be declared as such:

```
<references xmlns:qa="http://www.joker.playfiled.com/star-title/1.0"
           xmlns:pa="http://www.penguin.xmlli.com/review/1.0">
```

The declaration is associated with what is known as a URIs accompanying the prefix. The URLs are used to declare namespaces because they are guaranteed to be unique because domain names are registered in the World Wide Web therefore guarantees uniqueness.

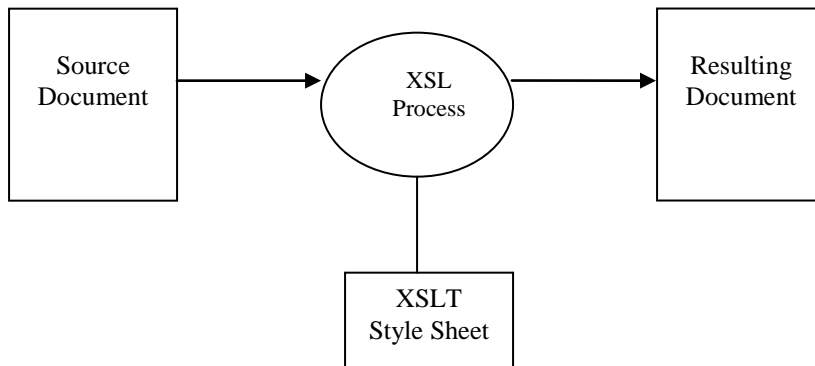
Final Sample:

```
<references>
  <rff: name
        xmlns:rff="http://catwoman.pineapplesoft.com/ref/1.5">
    MacMillan</rff:name>
  <link href="http://www.mcp.com"/>
  <ref: name
        Xmlns:ref="http://catwoman.apple.com/ref/1.4">
    AppleSoft </ref:name>
  <link href="http://www.applesoft.com/newsletter"/>
</references>
```

C. FORMATTING AND DISPLAYING XML DOCUMENTS

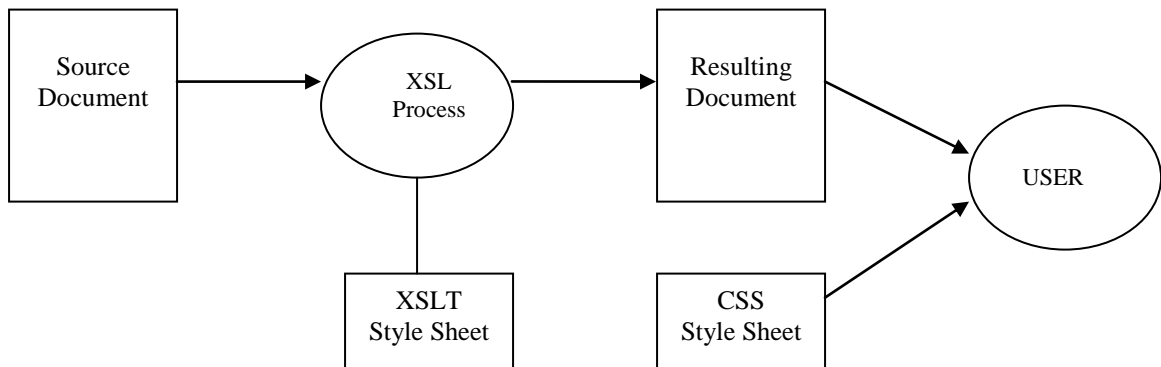
CSS, XSLT, and XSLFO

CSS or Cascading Style Sheets is a way for us to format HTML documents. XSL is the equivalent of CSS except it is for XML, but XSL is quite different from CSS. XML is all about structure, so XSL can format like CSS in ways of the style (font size, color, etc.), but it also formats in ways of structure like tables. There are two types of XSL: XSLT and XSLFO. In this paper, we will only discuss XSLT. Consider the following diagram for explaining XSLT:



XSLT

XSLFO takes XSLT further by allowing to combine the function of CSS and the functions of XSLT. Consider the following diagram for explaining XSLFO:



XSLFO

XLST SAMPLE

Consider the following example:

XML Document

```
?xml version="1.0" encoding="ISO8859-1" ?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
```

XSL Document.

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/W3C-xsl">
<xsl:template match="/">
  <html>
  <body>
    <table border="2" bgcolor="yellow">
      <tr>
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="CATALOG/CD">
        <tr>
          <td><xsl:value-of select="TITLE"/></td>
          <td><xsl:value-of select="ARTIST"/></td>
        </tr>
      </xsl:for-each>
    </table>
  </body>
</html>
</xsl:template>
</xsl:stylesheet>.
```

Add the following reference to the XML document

```
<?xml-stylesheet type="text/xsl" href="cd_catalog.xsl"?>
```

Final Output:

Title	Artist
Empire Burlesque	Bob Dylan
Hide your heart	Bonnie Tylor
Greatest Hits	Dolly Parton
Still got the blues	Gary More
Eros	Eros Ramazzotti
One night only	Bee Gees
Sylvias Mother	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
When a man loves a woman	Percy Sledge
Black angel	Savage Rose
1999 Grammy Nominees	Many
For the good times	Kenny Rogers
Big Willie style	Will Smith
Tupelo Honey	Van Morrison
Soulsville	Jorn Hoel
The very best of	Cat Stevens
Stop	Sam Brown
Bridge of Spies	T'Pau
Private Dancer	Tina Turner
Midt om natten	Kim Larsen
Pavarotti Gala Concert	Luciano Pavarotti

D. VALIDATING DOCUMENTS

There are 2 types of XML documents Well-formed and Valid. A well-formed document is restricted by syntactic rules. A Valid document goes a step further by restricting by syntactic rules and a structure described in an XML Schema.

Well-formed documents basically have does not contain DTD (or XML Schema) so the XML processor cannot checks its structure, therefore it only checks the syntax. A Valid document contains an XML Scheme or DTD so the XML processor can check its validity.

E. PROCESSING AND MANAGING XML DATA

PARSERS

“A parser is the most basic yet most important XML tool. Every XML application is based on a parser (Marchal, 2000).”

A parser⁵ is a program that stands between the application and the XML files. Without a parser XML files cannot be read or viewed. Parsers are written in very low-level programming. Therefore programmers do not waste their time developing parsers. Parsers are usually shipped or usually come with the software program. Like Internet Explorer 4.0 came out with 2 XML parsers. The simple definition of a parser basically makes sense of the XML code so that the right format and information is displayed in the medium (browser, XML tool etc.)

SAX & DOM

SAX & DOM are complementary of each other because they serve different needs. Document Object Model (DOM) is best suited for forms and editors while The Simple API for XML (SAX) is best suited for application-to-application exchange. The DOM parser is applied towards browser applications which was built to unify the object models in both Netscape Navigator and Internet Explorer.

SAX being event-based is more efficient than DOM because it doesn't have to create a tree of objects unlike DOM. SAX already processes the XML data line-to-line so you can say that it interprets the XML data directly. DOM first reads all the lines and creates its object tree before interpreting.

DOM has an upper hand over SAX since it was standardized by W3C. SAX on the other hand is not, but it is already being widely accepted as a de facto standard.

(Marchall, 2000)

⁵ Parser – comes from the world of compilers. In compilers, the parser is the one that reads and interprets the data (Marchal, 2000).

III. CURRENT STATE OF XML & REVIEW OF RELATED WORKS

In describing the technology behind XML it would seem like it already has established a permanent place in the computer industry and yet it's only in its infant stage of being version 1.0. In the following sections III to V, you will be able to see just how well established and well-embraced this technology has become already building its standards into other technology platforms.

A. WML (Wireless Markup Language)

Wireless or Wi-fi technology has become a thing of the future and it is now in our presence and is building itself as promising technology as we move from cluttered cables and plugs that will soon become a thing of the past.

XML was not surpassed in this technology upbringing it has also entered into its domain and promises itself to become a major part of its development. In WML we will witness the evolution of the XML and HTML being integrated as one – XHTML. As described in the introduction section (XML and HTML), XML is has come to birth due to HTML's shortcomings (Marchall, 2000). XHTML⁶ a hybrid of XML and HTML is well on its way as becoming a new standard.

WML is officially developed and maintained by the WAP Forum⁷. A specification developed by the WAP Forum defines the WML 1.1 DTD (Document Type Definition). The WAP Forum is supported by the cellular industry tycoons Nokia, Phone.com, Motorola and Ericsson.

If you have a cellular phone that is WAP-Enabled this means it contains a software called 'microbrowser'. This microbrowser fully understands and therefore has the capability to handle entities in of WML. WML is a markup language built with XML specification using the DTD Schema.

WML is popular today in the use of cellular phones and PDAs (Personal Digital Assistant)

⁶ XHTML – XHTML will be discussed in more detail in Section 5 – Future Trends and Research Issues in XML.

⁷ The Wap Forum - <http://www.wirelessdevnet.com/channels/wap/training/wml.html>

B. XML PROTOCOLS

In communications as we learned in Networking, there needs to be a standard in which both parties should agree on a standardized application format as to be able to communicate with each other. This means that the programs should agree on a communication platform. The most popular of these networking protocols is the TCP/IP protocol in which the Internet and emailing standards are built upon.

XML also needs a common platform in order to address the problem of “standardized application-to-application messaging⁸”. The challenge behind the XML protocol is to build the communication standard in the application-layer transfer protocol⁹. This protocol understood by platforms in software programs, machine communication and organization communications.

Such popular and standardized XML Protocols are as follows:

1. SOAP: Simple Object Access Protocol
2. XML RPC: A Remote Procedure Calling Protocol
3. ebXML: electronic business protocol
4. Microsoft Biztalk

C. Using XML with Database

As described by the second type of XML classification application the data application, XML is built to integrate the document with the application in which it can become almost seamless. Once XSLT is applied to the XML document, the structure becomes that of a relational database. Even so, XML doesn't stop there. XML has also built itself fully integrated with database applications such as SQL Server.

Such applications of XML are the XQL (eXtensible Query Language) and the newer and better version of XQL, XQuery. XQuery is currently applied to the latest developments of Microsoft SQL Server (Morrison, 2002).

⁸ Cover Pages 'W3C XML Protocols'- <http://xml.coverpages.org/xp-w3c.html>

⁹ Please review Network Layers defined by the IEEE Standards.

IV. APPLICATIONS OF XML

A. MICROSOFT & XML

Microsoft like all the other technology vendors have embraced the XML technology. In doing so, Microsoft is the leading manufacturer of XML Tools. From BizTalk to WordML, Excel to Microsoft Visual Studio.Net. These tools allow you to open and create XML documents as well as XML Schemas (XSD).

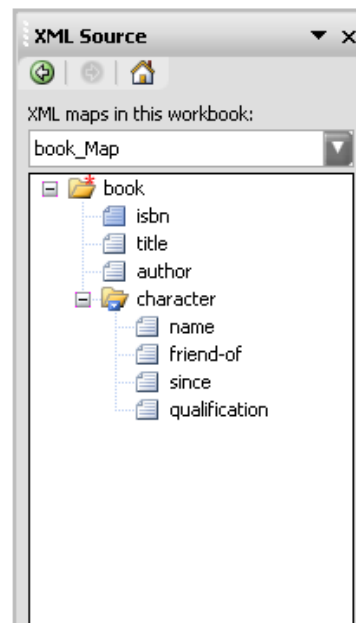
1. BizTalk: The Microsoft BizTalk Framework is an XML framework for application integration and electronic commerce. It is an industry initiative started by Microsoft and support by a wide range of organizations, from technology vendors like SAP and CommerceOne to technology users like Boeing and BP/Amoco.
2. Microsoft Word (WordML) & Excel (SpreadSheetML)
3. Microsoft .Net, Access, and Web Services
4. Microsoft XML Notepad
5. Microsoft SQLServer and XQuery

(St. Laurent, [No Date Specified])

```

<book >
  <title>Being a Dog Is a Full-Time Job</title>
  <author>Charles M. Schulz</author>
  <character >
    <name>Snoopy</name>
    <friend-of>Peppermint Patty</friend-of>
    <since>1950-10-04</since>
    <qualification>extroverted beagle</qualification>
  </character >
  <character >
    <name>Peppermint Patty</name>
    <since>1966-08-22</since>
    <qualification>bold, brash and tomboyish</qualification>
  </character >
</book >
  
```

XML Opened in Microsoft Word (WordML)



XML Opened in Microsoft Excel (XML Source Panel)

	A	B	C	D	E	F	G
1	isbn	title	author	name	friend-of	since	qualification
2	836217462	Being a Dog Is a Full-Time Job	Charles M. Schulz	Snoopy	Peppermint Patty	10/4/1950	extroverted beagle
3	836217462	Being a Dog Is a Full-Time Job	Charles M. Schulz	Peppermint Patty		8/22/1966	bold, brash and tomboyish

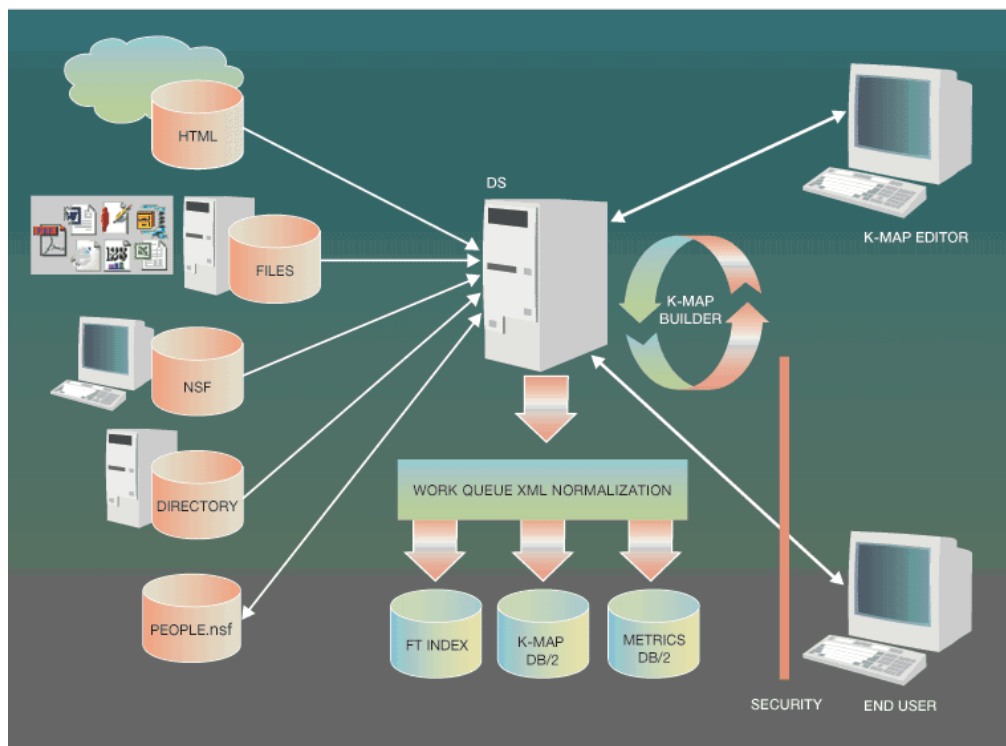
XML Opened in Microsoft Excel (SpreadSheetML)

B. IBM: THE LOTUS KNOWLEDGE DISCOVERY SYSTEM

The Lotus Knowledge Discovery System (LKDS) is a Knowledge Management System (KMS) that creates association between corporate experts and employees. It's main concern as of any KMS, is to disperse the tacit knowledge of the experts and be shared with the employees.

The LKDS makes use of what it calls 'Spiders'. These spiders are programs that collect the data throughout the system. It then brings the data to the 'Work Queue XML Normalization' where the data collected is filtered and organized. The spider extracts such data as: authors, usage, content, security and source location information (Pohs 2001).

Figure 2 The Discovery Server architecture includes both data and process components.



V. FUTURE TRENDS AND RESEARCH ISSUES OF XML

B. XHTML

If there was one thing to remember it is that XML was not developed to replace HTML as I mentioned in the first section of this paper, 'History or Evolution of the Technology'. It was developed to support HTML's shortcomings as the web became more and more demanding. XML was created because HTML was successful. Now let me introduce you to XHTML.

Over billions of web pages are found in the World Wide Web. Most of these pages are very unstructured, which often come unnoticed by the users. This is actually one of the most essential reasons why it takes so long for the browsers to download the pages, though very commonly the user blames this on the connection. It has come to be so horrendous that web browser vendors have had to create HTML processors to understand, read and display these unstructured and very messy codes. This is basically the result of poor coding, browser leniency, and proprietary browser extensions (Morrison, 2002).

The most obvious solution to this problem is to use XML. In using XML it just won't allow you to create pages that are unstructured, because the main concept behind XML is to develop structured data. But the question is how? As we know XML cannot replace HTML and there are just too many pages already developed in HTML that it would be impossible to replace all of them with XML structure. Today, these pages are already being called Legacy HTML documents. Just like legacy systems usually found in banks, it becomes very difficult for them to replace these systems. So even if XHTML is adopted by the web developers, web browsers will still need to be able to support HTML. "Over time, these legacy HTML documents could eventually be supplanted by valid, well-formed XHTML documents with plenty of structure (Morrison, 2002)."

B. Microsoft & Closed Source

The creation of XML was to build a structure that was simple, yet rigid in its creation so that it could be easily adopted by humans. But we are so far from this ideal, that an article mentioned this is slowly becoming more of a rumor than becoming a reality (Clark, 2003). But the hope of this still becoming a reality still has light at the end of the tunnel. And the light is Microsoft. Since 2002, when Microsoft opened its arms to XML by integrating it to so much of its new applications (St. Laurent, [No Date Specified]), the whole XML galaxy

came to a reality that this was real. At this point in time Microsoft is known to be the biggest and best supporter of XML Tools.

There's a big downside to this reality. As we look at Microsoft history, Microsoft is not a player of Open Source. So even though it has embraced what is known as 'Open Office' in which XML geeks are allowed to tinker with the code they are still constraint in using Microsoft Tools. So even if they do innovate, the tools used is still Microsoft's bread and butter so this will still be dominated by Microsoft.

The problem with this is, XML was created to be support Open Source. One can tinker with the code and pass it on. But with Microsoft's leading the competition in developing XML Tools that is very unlikely to happen.

REFERENCES

- Clark, Kendall Grant. "At Microsoft's Mercy".
<http://www.xml.com/pub/a/2003/04/23/deviant.html>. April 23, 2003
- Cover Pages. "W3C XML Protocol", Technology Reports.
<http://xml.coverpages.org/xp-w3c.html>. August 3, 2001.
- Foo, Soo Mee & Lee, Wei Meng. "XML Programming Using the Microsoft XML Parser", Appress. 2002
- Marchal, Benoit. "XML By Example", Que Publishing, 2000
- Mercer. "XML: A Beginner's Guide", Blind Folio. 2001
- Morrison, Michael. "Sams Teach Yourself XML in 24 Hours. 2nd Edition", Sams Publishing. 2002
- Pohs, Pinder, Dougherty & White. "The Lotus Knowledge Discovery System: Tools and Experiences", IBM Knowledge Management. Vol. 40, November 4, 2001.
- Refsnes, Jan Egil. "XSL-Transformation". XMLFiles.com
http://www.xmlfiles.com/xsl/xsl_transformation.asp
- St. Laurent, Simon. "Microsoft Office 2003 and XML".
<http://www.simonstl.com/articles/officeXML/index.html>
- The Wap Forum. "The Wireless Markup Language". Official Developer of WML.
<http://www.wirelessdevnet.com/channels/wap/training/wml.html>.
- Unknown Author. "XML Reference". DevX.com – The Know-How Behind Application Development.
<http://www.devx.com/projectcool/Article/20014>.
- Vlist, Eric van der. "Using W3C XML Schema", O'Reily XML.Com – XML from Inside Out.
<http://www.xml.com/pub/a/2000/11/29/schemas/part1.html?page=1>. October 17, 2001