

The Capabilities of XML & Its Interaction With Legacy Databases

Nigel McKelvey, B.Sc. (Hons).

Master of Science (M.Sc.)

Letterkenny Institute of Technology

**Ms. Ruth Lennon M.Sc. &
Mr. Thomas Dowling M.Phil.**

Submitted to the Higher Education and Training Awards Council, July 2003.

lyit | Institiúid Teicneolaíochta Leitir Ceannainn
Letterkenny Institute of Technology

Abstract

This thesis endeavours to establish which of the applications within the Extensible Markup Language (XML) available at present, meet the requirements of a modern business engaging in Electronic Commerce (E-Commerce) and/or the advancement of Web-based applications. The facilities provided by XML for improved network capabilities are also reviewed. There are numerous markup languages available at the time of writing. A review of six are provided in this document outlining the various advantages and disadvantages of each. Following this, XML applications have been identified that appear to provide superior networking and Web-advancement capabilities.

XML capabilities examined in more detail, include: connections to database systems such as Microsoft Access and Oracle 8i using Java Database Connectivity – Open Database Connectivity (JDBC-ODBC). An examination into how XML interacts with legacy data was carried out in relation to E-Commerce applications. At the outset it was clear that implications (such as such as interoperability issues) existed with these new technologies when interacting with legacy database.

Prototype systems were developed to highlight various issues regarding the capabilities of XML applications, for example file size reduction in Scalable Vector Graphics (SVG) and portability/browser support issues in Synchronous Multimedia Integration Language (SMIL) as well as XML's significant formatting requirements. Systems were also developed that demonstrate the ability of XML to interact with an Object Oriented Language such as Java. This interaction was investigated whilst connecting to a database incorporating the client/server architecture run locally on a Tomcat server revealing some JDBC-ODBC Middleware limitations for example. Issues such as interoperability and platform independence are discussed and documented. Finally, issues requiring further investigation are identified and recommendations outlined.

Acknowledgements

I would like to thank my supervisors, Ms. Ruth Lennon and Mr. Thomas Dowling for their dedication, guidance, pressure and unwavering support at all times.

I would also like to thank Mr. Liam Miller, Mr. Simon McCabe, Mr. Frederick Lusson and Mr. William Bennett for their guidance and time at various different stages of this M.Sc.

The time and effort put in by Library staff (especially Isabel) and Technical staff (especially Joe) are gratefully acknowledged and thanked, as are the respondents who took part in the questionnaire and assignment.

The support of my parents, family and friends was an invaluable asset to me constantly reviewing my progress and supporting my decisions.

Finally I would like to thank Michelle, for her support, loyalty and always being there.

lyit | Institiúid Teicneolaíochta Leictreonaíochta
Letterkenny Institute of Technology

Declaration

I hereby declare that with effect from the date on which this dissertation is deposited in the Library of Letterkenny Institute of Technology, I permit the Librarian of Letterkenny Institute of Technology to allow the dissertation to be copied in whole or in part without reference to the author on the understanding that such authority applies to the provision of single copies made for study purposes or for inclusion within the stock of another library. This restriction does not apply to the copying or publication of the title or abstract of the dissertation. It is a condition of use of this dissertation that anyone who consults it must recognise that the copyright rests with the author, and that no quotation from the dissertation, and no information derived from it, may be published unless the source is properly acknowledged.

lyit | Institiúid Teicneolaíochta Leitir Ceanaínn
Letterkenny Institute of Technology

Table of Contents

Declaration	i
Abstract	ii
Acknowledgements	iii
Table of Contents	iv
Table of Figures	ix
Table of Graphs	x
Table of Tables	xi
Table of Code Listings	xii
1 Introduction	
1.1 Introduction	1
1.1.1 Thesis Decomposition	3
1.1.2 Objectives of the Thesis	4
2 Technology Overview	
2.1 Development of XML	5
2.1.1 XML's Ancestry	
2.1.1.1 Advantages of HTML	8
2.1.1.2 Disadvantages of HTML	
2.2 XML Capabilities	10
2.2.1 XML as a Database	12
2.2.2 XML's Hierarchy	
2.2.3 Parsing	14
2.3 XML Business Capabilities	
2.3.1 Advantages of XML	15
2.3.2 Disadvantages of XML	16
2.4 Support for XML	17
2.5 Standard Generalised Markup Language	19
2.6 Document Object Model	21
2.6.1 Nodes	23
2.6.2 SAX	26
2.6.2.1 SAX Parsers	
2.7 SAX Versus DOM	28
2.7.1 Tree-based APIs – DOM versus Event-based APIs - SAX	
2.7.2 Incorporating the correct API	29
2.7.3 Advantages of SAX over the DOM	30
2.7.4 Advantages of the DOM over SAX	31
2.8 Document Type Definition	32
2.8.1 Cascading Style Sheet (CSS)	35
2.8.2 XML Namespaces	36
2.8.3 XML Schemas	37
2.9 XML Data Islands	40
2.10 Chosen Applications	41
2.11 Vector Markup Language	42
2.11.1 Advantages of VML	45
2.11.2 Disadvantages of VML	46
2.12 Scalable Vector Graphics	

2.12.1 Advantages of SVG	50
2.12.2 Disadvantages of SVG	51
2.12.3 Graphics Comparison	
2.13 Synchronised Multimedia Integration Language	52
2.13.1 Advantages of SMIL	56
2.13.2 Disadvantages of SMIL	
2.14 Mathematical Markup Language	
2.14.1 Advantages of MathML	64
2.14.2 Disadvantages of MathML	
2.15 Speech Markup Language	
2.15.1 Advantages of SpeechML	66
2.15.2 Disadvantages of SpeechML	
2.16 Voice Extensible Markup Language	
2.16.1 Advantages of VoiceXML	73
2.16.2 Disadvantages of VoiceXML	
2.16.3 Summary	
2.17 Data Transfers	74
2.18 XML and Databases	76
2.19 XSLT and XPath	78
2.20 Business-to-Business & Business-to-Consumer Applications	81
2.20.1 XML Web Services	
2.20.2 Legal Issues	85
2.21 XML and the Future	86
2.22 Conclusions	88
3 Designing the Prototype Software	
3.1 Introduction	91
3.2 System Requirements	
3.3 Design Specification	93
3.3.1 Scope of the Application	
3.3.2 Data Design	94
3.3.3 Designing the Interface	
3.3.4 User Interface Design Considerations	95
3.4 Hardware/Software Mapping	97
3.5 Global Resource and Access Control	
3.5.1 Relating Disability Issues to the System	98
3.5.1.1 Visual	
3.5.1.2 Auditory	99
3.5.1.3 Mobility	
3.5.1.4 Cognitive and Learning Disabilities	
3.5.2 Unicode Standard & Global Distribution	100
3.6 Subsystem Decomposition	101
3.6.1 Data Flow Diagrams	
3.6.2 Unified Modelling Language (UML)	104
3.6.3 Storing XML in a Relational Database (Oracle8i)	109
3.6.3.1 Available Storage Facilities	110
3.6.3.2 Loading Data into Oracle8i	111
3.6.4 Retrieving XML from a Relational Database (MS Access)	112
3.6.5 Data Transportation	
3.6.6 Structure of the System	113

3.6.7 Jackson System Development	115
3.6.8 Entity Relationship Diagrams	117
3.7 Relational Databases	119
3.7.1 Advantages of Relational Databases	120
3.7.2 Disadvantages of Relational Databases	
3.8 Java and XML	
3.9 Boundary Conditions	121
3.9.1 Within Boundary Conditions	122
3.9.2 Outside Boundary Conditions	123
3.10 Conclusions	124
4 Technologies Employed	
4.1 Introduction	128
4.2 Required Software/Hardware	
4.2.1 SVG Requirements	
4.2.2 VoiceXML Requirements	129
4.2.3 SMIL Requirements	131
4.2.4 VML Requirements	132
4.2.5 MathML Requirements	
4.2.6 Additional Requirements	133
4.3 Technologies Employed	134
4.3.1 Oracle8i	
4.3.1.1 Three-tier Client/Server Architecture	
4.3.1.2 Advantages of Three-tier Client/Server	135
4.3.1.3 Disadvantages of Three-tier Client/Server	
4.3.2 Microsoft Access 2000	136
4.3.2.1 Creating Data Source with ODBC	
4.3.3 JDBC	139
4.3.3.1 JDBC-ODBC Bridge	140
4.3.3.2 Advantages of JDBC-ODBC	141
4.3.3.3 Disadvantages of JDBC-ODBC	
4.3.4 Tomcat Version 3.2	142
4.4 Alternatives to Java	
4.5 Conclusions	143
5 Testing and Evaluation	
5.1 Introduction	144
5.2 Introduction to Testing Methodologies	
5.2.1 Heuristic Methods	
5.3 Systems Evaluation	146
5.4 SMIL System Evaluation	
5.4.1 GUI Evaluation	
5.4.2 SMIL System Functionality	147
5.5 VoiceXML System Functionality	149
5.6 SVG GUI Evaluation	150
5.7 The Viability of XML Applications	151
5.8 Colour Functionality	153
5.9 Oracle System Timing Issues	154
5.10 MS Access System Functionality	157
5.11 Questionnaire Objectives	159

5.12 Questionnaire Design Guidelines	160
5.12.1 Likert Scale	161
5.12.1.1 Advantages of the Likert Scale	162
5.12.1.2 Disadvantage of the Likert Scale	
5.13 Conclusions	163
6 Summary & Conclusions	
6.1 Preface	165
6.2 Summary	
6.3 Conclusions	167
6.3.1 XML is a Database in itself	
6.3.2 Oracle is difficult to implement	168
6.3.2.1 Java and XML can successfully interact with a database	
6.3.3 JDBC-ODBC Middleware Limitations	
6.3.4 XML requires significant formatting	169
6.3.4.1 DTDs and XML Schemas provide a similar function	
6.3.5 File Sizes	170
6.3.5.1 SVG versus VML	
6.3.5.2 SVG Improves Bandwidth	
6.3.5.3 Text displayed using SVG appears ‘smudged’	171
6.3.5.4 SVG as an alternative to current technologies	172
6.3.5.5 SVG and VML can provide further alternatives	
6.3.6 Incorporating SMIL requires media considerations	
6.3.6.1 SMIL can be portable	173
6.3.6.2 Applying the correct bitrate in SMIL is important	
6.3.7 VoiceXML is a viable option for CBTs	
6.3.7.1 SALT will enhance Web applications	174
6.3.8 MathML reduces file sizes	
7 Recommendations for Further Research	
7.1 Recommendations	175
7.1.1 Further development of the JDBC-ODBC Prototype	
7.1.2 Further development of the Oracle Prototype	
7.1.3 Further development of the VoiceXML Prototype	
7.1.4 Research into the Advancement of GIS with SVG	176
7.1.5 Research into the capabilities of SVG for rendering photograph quality images	
7.1.6 Research into XML’s capabilities for an improved Internet	
7.1.7 Research new/other XML technologies	177

Appendices

Appendix A	References	i
Appendix B	Bibliography	xi
Appendix C	Glossary/Nomenclature	xiii
Appendix D	Acronyms	xxiv
Appendix E	Code Keys	xxvi
Appendix F	Questionnaires/Test Data	xxix
Appendix G	Applications Running	xli
Appendix H	Performance Measurements	lii
Appendix I	Program Listing	lvii
Appendix J	Publications	clxxv

lyit | Institiúid Teicneolaíochta Lettir Ceanáin
Letterkenny Institute of Technology

Table of Figures

2.1	XML's Ancestry	6
2.2	XML Hierarchy	13
2.3	An XML System	18
2.4	The tree representation of an HTML document	23
2.5	DOM Level 2	25
2.6	XML Parsing	28
2.7	Syntax Example for Handling a Callback Event Handler	29
2.8	Syntax to Illustrate Event-Based API	30
2.9	Tree representation of Code Listing 2.4	34
2.10	VML Structure	45
2.11	SVG Layout	47
2.12	Conversion of SVG Using Batik	48
2.13	A typical SVG Viewer	49
2.14	SMIL Elements within a Presentation	54
2.15	Telephone Interaction with a VoiceXML Interpreter	68
2.16	Markup Language at a systems-level	72
2.17	Three-tier architecture for a Web database system	77
3.1	DFD – Interacting with a SMIL Presentation	102
3.2	DFD – Verifying a Credit Card with VoiceXML	103
3.3	UML State Diagram: Selecting a Menu Option in an Oracle Database	105
3.4	UML Dynamic Model: Adding a Name Into an Oracle Database	106
3.5	UML Use-Case Diagram: Inputting Details Into Oracle	106
3.6	UML Business Process Model: Viewing Details in MS Access	107
3.7	Class Diagram - XML document being uploaded into Oracle	108
3.8	Accessing the JDBC for Oracle	109
3.9	DBMS Architecture	110
3.10	System Structure Diagram	114
3.11	JSD Diagram depicting the main operations within the SVG page	116
3.12	Screen shot of the 'SVG Page'	117
3.13	ERD – Ordering a Tutorial	118
3.14	ERD – Entering Customer Details to the Database	118
4.1	Three-tier Client/Server Architecture	135
4.2	Data Source Administrator Options	137
4.3	Microsoft Drivers	137
4.4	Entering the DSN	138
4.5	Servlet handling a request	139
5.1	The Software Testing Process	119
5.4	Updating in an Oracle System	156
6.1	Visual Differences When Rendering Text	171

Table of Graphs

5.1	User Satisfaction with SMIL Functionality	148
5.2	Respondents that preferred XML to SVG for Text Displays	150
5.3	Average Connection Times (Oracle) - 50 XML Files In The Database	155
5.7	Increased Data Load – Query Execution Times (Oracle System)	157
6.2	File Size Comparisons For Rendering Maths	174

Table of Tables

2.1	Data Elements	9
2.2	Support Available for XML Browsing	17
2.3	DOM Interfaces	22
2.4	Freely available SAX Parsers	27
2.5	Media Support in SMIL	53
2.6	Support for MathML	58
2.7	VoiceXML versus SALT	74
2.8	Products and Services	84
3.1	Key to Figure 3.10	115
3.2	Book Details represented in a table	119
3.3	Sigma-X: Tutorial Demonstration/Application of XML	122
3.4	Omikron-B: Uploading XML into Oracle8i	123
3.5	Kappa-C: VoiceXML Capabilities	123
4.1	Technologies Implemented	128
4.2	SVG Requirements	129
4.3	VoiceXML Requirements	130
4.4	SMIL Requirements	131
4.5	VML Requirements	132
4.6	MathML Requirements	133
4.7	Additional Requirements	133
5.1	XML Applications – GUI Evaluation	146
5.2	XML Applications – Browser Test	153
5.3	Data Types returned as XML from MS Access via JDBC-ODBC	158

Table of Code Listings

2.1	Markup of a Library	10
2.2	Sample HTML code	22
2.3	A well-formed XML Document called Book.xml	33
2.4	A DTD called Library.dtd	34
2.5	CSS Code	35
2.6	Applying a Namespace to an XML file	37
2.7	XML Schema Example	38
2.8	Applying an XML Schema to an XML File	39
2.9	Example of an XML Data Island	41
2.10	VML Code to display a Red Square	44
2.11	SVG Code to display a Red Square	50
2.12	Displaying various media elements using SMIL	54
2.13	MathML Markup	59
2.14	Displaying MathML using XHTML	62
2.15	Displaying MathML in Netscape 6.2	62
2.16	SpeechML Code	66
2.17	Altering the pronunciation of a word in VoiceXML	68
2.18	VoiceXML Code	69
4.1	Embedding a SMIL Presentation into an HTML Page	132
5.1	Putting a timer into Java	155

Chapter 1

Introduction

lyit | Institiúid Teicneolaíochta Leitir Ceanaínn
Letterkenny Institute of Technology

1.1 Introduction

A new capability rapidly growing in popularity is that of data exchange across the Internet. It is in this environment that XML's capabilities are being realised. XML is touted as the next-generation Web language that defines how data can be transmitted and shared across the Internet. It is also becoming 'the common language' both for E-Commerce and for applications integration [LSGX]. It is for this reason that many software developers are investigating the feasibility of developing applications that encompass XML or applications of XML.

Data sent across the Internet can come from various sources. One such source is legacy databases (older data, which can be stored hierarchically and can also be used to extract trends etc.). In order to transport this data across a network, Middleware may be required to enable more flexible Client/Server interactions. This thesis also investigates the capabilities of XML when interacting with these legacy database systems. The viability of creating XML documents from this legacy data was also investigated. The technology essential for the support of the server side of these applications, often involves intensive interaction with legacy databases. The implications of the interactions of XML with legacy database systems may have many undocumented repercussions. It is for this reason that it was deemed necessary to research these topics. Applications built for operation across the Internet are also increasingly developed using Java. Both Java and XML have the advantage of being platform independent thus enabling applications developed through these languages to run on any operating system. Thus, it was decided to develop a prototype application encompassing both XML and Java to test portability issues involved in such applications.

With any Web development strategy, striving to reduce file sizes is always an important aspect as well as improving bandwidth capabilities. Therefore, this thesis investigates the capabilities of XML (and applications of XML) with regard to improved Internet efficiency.

There are numerous sources such as books/journals, etc., which consider issues regarding XML's potential and issues raised by XML software developers. However, research into XML's capabilities is to a large extent primarily focused on one XML technology at a time as opposed to a system that encompasses several XML technologies. Therefore, much of the research into this area may be considered limited. There has not, as yet, been any significant research into the possibility of merging several applications of XML into one application and testing its capabilities. It is the intention of this thesis to provide an outline for such research as listed below:

- To investigate the interaction of XML with database systems (i.e. Oracle and MS Access), test the interaction and document any repercussions.
- To research the capabilities of XML and applications of XML such as SMIL, SVG, VML, MathML and VoiceXML.
- To investigate the capabilities of XML across networks (i.e. file size reductions, client/server transformations, formatting etc.).
- To investigate the impact of XML on Web site advancements with particular reference to information display/presentation.

The speed at which XML technologies are growing is exponential. Therefore, this thesis will serve as an up-to-date reference for any future developments in the area.

1.1.1 Thesis Decomposition

The document itself is divided into chapters that serve to describe/demonstrate the capabilities of XML based on literature findings and practical findings as outlined below:

Chapter 1: The function of this chapter is to provide a background for the chosen topic. The document structure is also outlined with the primary objectives of the thesis also summarised.

Chapter 2: This chapter serves as a Literature Review of the technologies outlined through books, journals, papers and reports. The capabilities of XML are outlined in this chapter.

Chapter 3: The purpose of this chapter is to provide an overview of how the prototypes were designed. The chapter includes Unified Modelling Language (UML) diagrams, Entity Relationship Diagrams (ERDs), Data Flow Diagrams (DFDs), and Jackson diagrams, etc. The chapter also discusses the importance of good design and specifies certain aspects of the system(s) that warranted particular attention (e.g. functionality, Human Computer Interaction (HCI), etc.).

Chapter 4: This chapter outlines the various different technologies that were implemented/required in order to run the prototype application(s) successfully. Hardware and software requirements are outlined with tables indicating how these technologies can be obtained, their cost, suppliers, etc.

Chapter 5: The main objective of this chapter is to present an analysis of the results of the testing and evaluation procedures implemented. The system(s) were tested for functionality and GUI requirements.

Chapter 6: This chapter summarises the research conducted based upon literary and practical findings. It restates the issues investigated and the findings of these investigations. This chapter also presents a series of conclusions based on the research conducted. It is from these conclusions that recommendations are extracted.

Chapter 7: This chapter indicates what further work could/should be conducted in order to further investigate issues that were outside the scope of this thesis.

1.1.2 Problem Definition

A literature survey was conducted which highlighted two main ‘problems’ that warranted investigation. Therefore it was necessary to create system(s) that would help evaluate XML as a technology. The main problems included:

- The interoperation of XML with legacy databases.
- The capabilities of XML and applications of XML.

Chapter 2

Technology Overview

lyit | Institiúid Teicneolaíochta Leitir Ceanaínn
Letterkenny Institute of Technology

2.1 Development of XML

The Extensible Markup Language (XML) is a rapidly growing technology with many capabilities, which are discussed in this thesis. In order for XML to be researched and properly understood, it is necessary to first discuss how and why XML was developed. Before its creation, Hypertext Markup Language (HTML) was the primary means of displaying data on the Internet and, for HTML to exist the Standard Generalised Markup Language (SGML (ISO 8879)) needed to be in place in order to define the grammar. SGML can be more complex to use than XML and does not have the same level of industry support as XML. These are just a few aspects that led to the development of XML, which will be discussed in more detail later in this chapter (Sections 2.3 and 2.4).

2.1.1 XML's Ancestry

A markup language may be described as a language that provides 'information about information' or meta information. Figure 2.1 depicts the ancestry of these markup languages. Generalised Markup Language (GML) is the basis on which HTML was created and, therefore, is to a large degree responsible for many of HTML's strengths and weaknesses. GML was developed in 1969 by Charles F. Goldfarb, Ed Mosher and Ray Lorie in order to solve the data representation problem [GMLC]. As time went by the need for a validating parser that would read a Document Type Definition (DTD) and ensure the accuracy of the markup became clear [GMLC]. SGML was deemed to be a suitable solution. SGML is a formal system designed for building text markup languages (refer to Section 2.5).

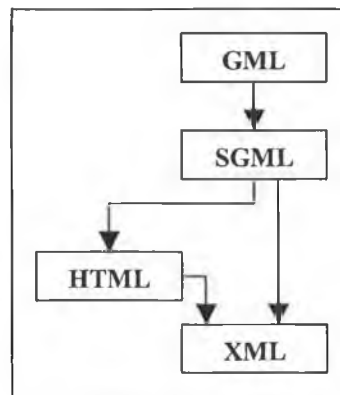


Fig. 2.1 XML's Ancestry [JVCB]

HTML is considered to be the basis for publishing hypertext on the World Wide Web. It is a non-proprietary format based on SGML, and can be created and processed by a wide range of tools, from simple text editors to sophisticated 'What You See Is What You Get' (WYSIWYG) authoring tools. However, the limitations of HTML (refer to Section 2.1.1.2) have fuelled the call for a technology such as XML. XML permits an enterprise to define its own markup language with particular emphasis on specified tasks, such as Electronic Commerce (E-Commerce), supply-chain integration, data management and publishing [XJAV]. If an enterprise can create their own markup language, then certain rules/regulations should be established.

The World Wide Web Consortium (W3C) is a working group that is known for its role as the keeper of the HTML standards. Its creation was necessary in order to act in the administrative role of managing the advancements in the technologies and ensure that the new languages being created would conform to a certain standard. The W3C was a venture by the MIT Lab for Computer Science in cooperation with INIRA (France) and Keio University (Japan) [XEBC]. The organisation employs technical staff, that helps to ensure the quality of results [XEBC]. These staff are necessary in order to ensure that XML's rules are adhered to.

The version of HTML that Tim Berners-Lee¹ invented was strongly based on SGML, an internationally agreed upon method for marking up text into structural units such as paragraphs, headings, list items and so on. SGML could be implemented on any machine. The proposal was that HTML was platform independent of the browser or other viewing software, which displayed the text on the screen.

Certainly the simplicity of HTML, and the use of the fixed element `<a>` for creating hypertext links, made Berners-Lee's invention very useful. In September 1991, the WWW-talk mailing list began, which was an electronic discussion group in which interested parties could exchange ideas. By 1992, other academics and computer researchers were beginning to take an interest.

XML documents follow a more stringent set of rules than HTML. Every open tag must have a corresponding close tag and "empty" tags may use a shorthand form (`<.tag/>`) [DTDG]. The use of pairs of tags such as `<title>` and `</title>` is taken directly from SGML. The SGML elements used in Berners-Lee's HTML included *p* (paragraph); *h1* through *h6* (heading level 1 through heading level 6); *ol* (ordered lists); *ul* (unordered lists); *li* (list items) and various others. By basing HTML on SGML people were less likely to create their own languages and this meant that the HTML was the primary means of displaying information on the Internet.

HTML is a fixed tag² set and it only describes documents of a single type. HTML documents attempting to function like applications are congesting the Internet with client-to-server traffic. Web pages that contain too many graphics and inefficiently formatted HTML send excessive quantities of data through the network. Large

¹ Invented the World Wide Web in 1989.

² HTML tags are usually English words or abbreviations but they are distinguished from the normal text because they are placed in small angle brackets. They essentially tell the browser what to do.

graphics should normally be avoided in Web pages but if they are unavoidable, techniques such as caching and compression can be used to reduce the flow of this binary data through the network. XML could change all of this. However, it is unlikely that all the useful HTML pages that exist will become redundant within the near future.

With regard to HTML formatting, there are many different ways to format the HTML syntax to produce the same results. Techniques such as Cascading Style Sheets (CSS) (refer to Section 2.8.1) can apply a certain look to a web page so that the HTML code can be less burdened with formatting and more focused on content. However, another approach is being taken in the form of XML. Critical appraisals of HTML have led to support for developing a technology such as XML [UPI]. HTML's limitations have brought about the call for and interest in XML technologies [HTMLF]. To summarise:

2.1.1.1 Advantages of HTML

- A wide range of tools can produce HTML.
- The simplicity of HTML, and the use of the fixed element `<a>` for creating hypertext links, was what made Berners-Lee's invention so useful.

2.1.1.2 Disadvantages of HTML

- HTML is a fixed tag set. It only describes documents of a single type.
- HTML documents that try to function like applications are clogging the Internet with client-to-server traffic. [HTMLF]

XML is a markup language for documents incorporating structured information [DDJH]. Structured information contains both content (words and pictures) and an indication of the functionality of the content. With XML technologies, the content and presentation are separated which makes XML a powerful form of data storage and a data descriptor [ISTS]. The data is tagged with an Extensible Stylesheet Language (XSL) document. The browser helps translate the XML data into HTML using the XSL document. This way, only the essential data is passed through the network multiple times. The XSL document resides cached in the user's browser and is used to translate the small XML document into a large HTML page or into another format [PXSL]. XML is a powerful and portable language as implementations of XML applications can be transformed using XSL [ACMK]. XML is designed to be a subset of SGML, in that every XML document should also conform to SGML standards [BTBF]. SGML was designed to give information managers more flexibility to say what they mean, and XML brings that principle to the Web. It could be argued that data elements stored within a database have no meaning [DBMSB]. It is the relationship between data elements that when presented in a given format, provides information. The following data elements for example could be kept in a data store:

Book ISBN	Book Title
Author	Subject Area

Table 2.1 Data Elements

When these attributes are separated they hold no meaning, however, if they are assembled together in a structured form then these attributes, which relate to an entity, form information [DBMSB]. In the example given below the document could be a

Library and the information provided would aid the correct filing of the books. The following is a markup of the data elements described in Table 2.1:

```
<Library>
Book_ISBN="5674534200"
Book_Title="A very good book"
Author="Joe Bloggs"
Subject_Area="Java and XML"
</Library>
```

Code Listing 2.1 Markup of a Library

There are various ways to utilise XSL in web applications such as server-side transformation and client-side transformation. However, the technique that puts the least burden on the web server is the client-side transformation approach [PXSL]. This is because the web server does not need to format the data into a large HTML file and also does not need to send this large HTML file through the network.

Microsoft's Vector Markup Language (VML) is an XML application for vector graphics that can be embedded in Web pages in place of the Graphics Interchange Format (GIF) and Joint Photographic Experts Group (JPEG). XML can provide a means of successfully incorporating multimedia onto the web as it can reduce the size of these files, which will be discussed later in this chapter.

2.2 XML Capabilities

XML provides an environment for the integration of unrelated forms of data in a platform that is independent of other languages [DXDB]. This data can include varying types of content. That is, content in a section heading has a different meaning from content in a footnote and so forth. Most documents have some structure. XML

provides a standard format for the description of data thus allowing applications to retrieve data from a variety of web sites and subsequently to assimilate the data. A markup language is a method of identifying structures in a document. The XML specification defines a standard way to add markup to documents.

XML was created so that highly structured documents could be more efficiently accessed over the web. An example of such documents could include the use of a Synchronised Multimedia Integration Language (SMIL) application. SMIL documents are derived from XML and allow the synchronisation of various media elements such as text, sound, video, etc. over the Internet. Possible alternatives include, HTML, SGML, and XSL. However these alternatives may not necessarily be practical for this purpose [AXML]. A full SGML system, which provides arbitrary structure, can resolve large, intricate problems that validate their expense. Viewing structured documents sent over the Internet seldom carries such validation.

At St. John Health System Hospital in Tulsa, a project was constructed in 1998 which was a cohesive, hospital-wide database containing all patient records, including X-rays and other images, which is currently accessible to any doctor with access to the St. John MedWeb intranet [XMDH]. The hospital has different databases scattered across departments ranging from administration to radiology. In 2000 XML was utilised as an enabling technology for St. John's to create a unified database from diverse types of data in dissimilar file formats, which would aid improved querying.

This practical study supports Kendall's³ statement that,

*"XML technology will enable us to tie images and transcripts..." into
"...one seamless package" [XMDH].*

³ Senior vice president at St. John's

2.2.1 XML as a Database

In most cases a database is considered to be data stored in tables, presented in forms and manipulated through queries. The definition for a database is, 'a collection of data' [DBMSB]. An XML database has to deal with informational documents and data elements [DBMSB]. This is considered to be similar to a content management system and a relational database working alongside each other in an integrated system. Therefore, it can be suggested that an XML document is a database in itself as it too is a collection of data.

XML databases will have an increasingly important role to play within the business framework [DBMSB]. Therefore, it is of paramount importance that standards are defined and that integrating XML into the business framework is made easier by conforming to those standards.

2.2.2 XML's Hierarchy

XML, which is somewhat similar to HTML, is a subset of SGML, which is often used to format technical manuals, archival records and other large documents. Whereas HTML is used to define the appearance of a document on the Web, XML is used to identify the data inside it. For example, an XML-based form might utilise tags such as `<colour>`, `<firstname>` and `<lastname>` to identify the various data fields and to store XML information. By using such tags, it becomes easier to differentiate between the word "brown" as a last name and "brown" as a hair colour.

Information contained in XML documents is stored in a hierarchical approach as can be seen in Figure 2.2. Everything in the structure relates to each other in some way either through parent/child or sibling/sibling relationships [BXML].

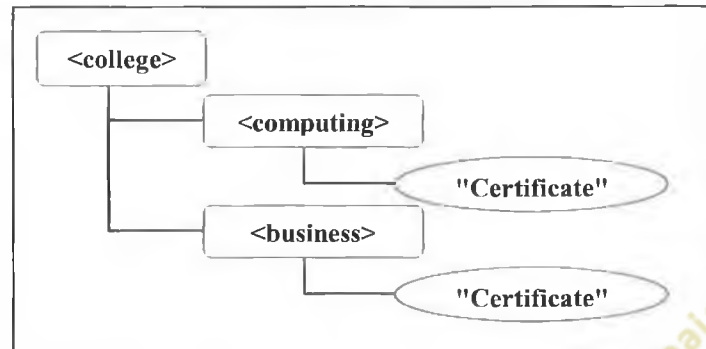


Fig. 2.2 XML Hierarchy

In Figure 2.2 *<college>* is the parent with its children being *<computing>* and *<business>*. *<computing>* and *<business>* are siblings to each other with the text in inverted commas being their “children”. Understanding this structure makes working with XML a lot easier and makes introducing this method to other technologies more acceptable.

Another advantage of XML is that it is, as its name implies, eXtensible, meaning that users can formulate new data tags at will without having to worry whether or not other XML applications will be able to interpret them. A new tag is simply treated as a new category of data.

XML incorporates both client-side and server-side utilities. On the client, it is possible for XML to play a role inside browsers. An XML-enabled browser, for example, can offer improved querying and search capabilities for Web sites incorporating XML. This is an aspect of XML that could be further researched and developed.

2.2.3 Parsing

Parsing is an aspect of XML that could possibly provide a basis for improved querying across networks (refer to Page 28 for further analysis of XML Parsing). On the server, XML may be used as a middleware link between differing data sources. Coco Jaenicke, product-marketing manager for Object Design, makes the point that XML will be most beneficial as a way to translate data from different legacy databases and applications. She stated that:

"XML acts as a common denominator for all data formats."

[XMDH]

2.3 XML Business Capabilities

XML is a technology that has been gradually developing over the past few years. Due to its nature, this technology has often been employed in the fields of E-Commerce, Business-to-Business (B2B) and Business-to-Consumer (B2C) (refer to Section 2.20). One of the main reasons for the development of XML was that an objective of E-Commerce was to enable companies on different platforms to communicate with each other [CCMI]. Businesses communicate with customers and partners through various channels. The Internet is one of the most recent and, for many businesses it is an important communications channel. It is fast, relatively reliable, reasonably priced, and universally accessible [ECOM]. The technology essential for the support of the server side of these applications, often involves intensive interaction with legacy databases. The implications of the interactions of this new technology with legacy database systems may have many undocumented repercussions. This is the main reason why this thesis has been written. Managers and entrepreneurs are trying to find a way to maximise their business potential on the Internet [BXML]. XML provides

these companies and organisations with a means of providing a service to a worldwide audience.

One of the most famous companies around today is *Amazon*. Services have gone through rigorous development and debugging so as to provide a secure service available to customers. Credit card numbers are encrypted in a way that human intervention is not necessary, thus increasing security. *Amazon* may also be seen to provide B2B services as various businesses may purchase books from Amazon at www.amazon.com [BXML]. In March 2003, their web site revealed that the American Customer Satisfaction Index (ACSI) issued Amazon.com with a score of 88 (which is up 5% on the previous year). This suggests that E-businesses are growing in popularity. E-Business sales of airplane tickets have also increased dramatically in recent years

The fact that so many organisations had legacy data stored in back-end systems drove the development of XML [CCMI]. XML could become a leader in the Electronic Data Interchange (EDI) market, as well as in many expanding markets, such as financial services, manufacturing and health care, which have large legacy databases that need to be integrated with new applications and data sources.

2.3.1 Advantages of XML

- The filtering of data implies that unnecessary data is ignored and only data that is required is passed through the network.
- XML provides an environment for the integration of unrelated forms of data in a platform that is independent of other languages.

- XML provides a standard format for the description of data thus allowing applications to retrieve data from a variety of web sites and subsequently to assimilate the data.
- Highly structured documents can be accessed over the web.
- Users can formulate new data tags at will without having to worry whether or not other XML applications will be able to interpret them. However, a namespace (refer to Section 2.8.2) should be applied to these new tags in order to make it unique and to avoid two tags having the same name but describing different data.
- XML incorporates both client-side and server-side utilities.

2.3.2 Disadvantages of XML

- The implications of interactions of this new technology with legacy database systems may have many undocumented repercussions (refer to Chapters 5 and 6).
- As XML is relatively new, support for its capabilities is to some extent limited.
- As a result of the variety of different vendor versions of XML being developed there is a possibility that they will not all work together over the Web. Certain aspects of the language that make it effective (meta data) pose potential security problems. For example, vendor specific versions may prove incompatible. [CCMI]

2.4 Support for XML

Accessing data stored in XML documents and displaying the contents through style sheets is an important aspect of XML's capabilities. However, if browser support is not sufficient then problems can arise (these are discussed throughout the remainder of this chapter). As of 2002 Microsoft's Internet Explorer 5 provides complete support for the DOM and XSL. In contrast, Internet Explorer 5 does not provide complete support for the CSS Standard, which is arguably the best means of displaying data in a browser (refer also to Chapter 5, Table 5.2). Table 2.2 [BWXL] lists the support for XML browsing currently offered (2nd May 2000):

	Netscape 6 (release 1)	Opera 4 beta 3	IE 5.0/Mac	IE 5.5/Win
XML Display without style sheet	Everything as inline	Everything as inline	Structured Presentation	Structured Presentation
Uses W3C PI to connect style sheet	Yes	Yes	Yes	Yes
Display:inline	Yes	Yes	Yes	Yes
Display:block	Yes	Yes	Yes	Yes
Display:none	Yes	Yes	Yes	Yes
Display:list (and list-item)	Yes	Yes (numbering issues)	Yes (numbering issues)	No
Display:table (and contents)	Yes	Yes (some interactions with embedded HTML)	No	No
XLink Support	Yes (3/3/98 draft)	Through CSS Extensions	No	No
Embedded HTML Support	Yes (using HTML 4.0 URI namespace)	Yes (using HTML 4.0 URI as namespace, some problems with HTML <i>img</i> elements)	Yes (using html prefix, some problems with HTML a elements)	Yes (using html prefix)
XSLT Support	No	No	Yes, based on old draft	Yes, based on old draft

Table 2.2 Support Available for XML Browsing

Figure 2.3 shows how a typical XML systems operates:

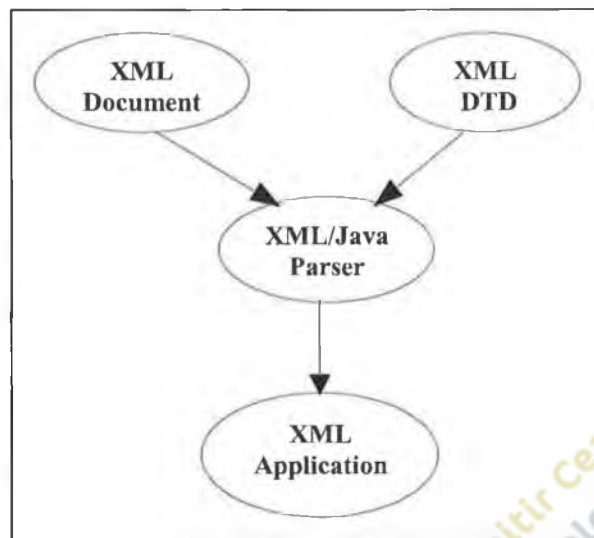


Fig. 2.3 An XML System [AXML]

Figure 2.3 shows how the XML Document and the XML DTD (refer to Section 2.8) pass their information to the parser. The parser used to obtain this information is simply a program that can read XML syntax. By using a parser the finished application will never have to look at the XML for the information, thus speeding up the display process. The parser is regarded as an integral part of a Java-XML combination. Programmers have to access data in XML documents through the operations of a Java XML parser [SAXI]. Two kinds of parsers include: Simple API for XML (SAX) and DOM parsers. The SAX and DOM Application Programming Interface (APIs) both allow access to information stored in XML documents through programs created in Java for example [SAXI]. The SAX and DOM APIs are very different (refer to Sections 2.7.1 and 2.7.2). SAX is well suited to allowing programs to read in information that is data generated by software [SAXI]. DOM on the other hand is better suited for reading in information that is stored in documents.

The uses of XML can be split into two categories:

1. The mark-up of text documents and
2. the representation of raw data. [DBMSB]

To summarise, XML offers the opportunity to look at information through more than one view, giving the technologies discussed in this document a wide range of possibilities for further development. Nonetheless, for these possibilities to be realised certain rules must exist that will govern the behaviour of the code and how that code can be implemented.

2.5 Standard Generalised Markup Language

The Standard Generalised Markup Language is used to describe the content of text documents in a logical and structural manner. SGML is occasionally referred to as a metalanguage⁴, because it includes rules for writing other markup languages [XEGP]. SGML does not define a default language. However SGML users must define or utilise one of the subset languages to create SGML documents. Each of these languages is based on a DTD. In addition SGML includes rules for the behaviour and use of these languages. HTML is defined as an SGML markup language. It is the best-known SGML markup language but is mistakenly considered to be a part of SGML. XML is not a subset markup language. It performs the same general function as SGML. However it does not include a lot of the features of SGML and as a result its use is subsequently promoted on the Web [XEGP].

⁴ A language used to write other languages

SGML differs from other document formats because it uses structural context rather than formatting characteristics to describe document content. SGML languages define descriptive content containers that contain document content and are described in relation to other containers in terms of where they can and should exist. Document authoring and formatting rely on the structure of SGML documents to guarantee that content is accurately described and formatted. Structural trees can be built from SGML documents, visually rendering their structure and illustrating their dependence on the content (refer to Figure 2.2). SGML further differs from other document formats by defining structure, content and style separately, allowing document designers to focus on each document component separately. This then reduces the size, distribution and maintenance costs of documents. SGML is used principally for document creation, storage, and distribution and as a source format for conversion to other document formats. SGML documents can be read and manipulated by software such as browsers and authoring tools because of the high level of content description. Descriptive content containers are called Elements and are often referred to as Tags. Elements are usually given clear English-based names that are based on their function or content, such as *paragraph*, *table*, *warning* or *graphic*. The name for paragraph elements, for example, is often *p*, *para* or *paragraph*, but could be anything conceivable, for example TeXT. Document authors describe documents with these descriptive elements, producing well-described SGML documents. Descriptive, clear and explicit element names are a requirement for effective SGML document production, simplification and reduction of the cost of stylesheets and software development [XEGP].

Online learning applications can use SGML for the creation, maintenance and storage of their documents [XEGP]. SGML allows course authors to write and store their courses in one or more SGML documents, depending on the manner in which their SGML system is configured. The same activity in HTML would require authors to produce hundreds and possibly thousands of HTML pages, depending on the size of their course. The reason for this difference is because SGML has the ability to describe any organisation of data, while HTML is limited to Web pages. As a result of some of the aforementioned differences, SGML systems are more efficient, easier to maintain and more cost effective than their HTML counterparts. For example, SGML-based courses can be delivered to print or Web targets without difficulty, whereas HTML is simply not designed to exist outside of Web browsers.

2.6 Document Object Model

A Document Object Model (DOM) is an Application Programming Interface (API) for representing a document (such as a HTML document – see Code Listing 2.2) and accessing and manipulating the different elements (such as the HTML tags and strings of text) that encompass that document. A set of APIs are declared using the DOM and these APIs set out how data can be accessed and manipulated by the application within a document. Other means of programmatically manipulating data in an XML document include Extensible Pointers (XPointers) and Extensible Query Language (XQL) [SDPPO]. The DOM creates a partition between the XML parser and the applications themselves [DBMSB]. This is necessary because the DOM has an interface (refer to Table 2.3) that programs use for manipulations and it also has a set of APIs that define how an application can carry out these manipulations. For this reason it is necessary to separate these aspects. The DOM comes with a set of optional

interfaces that enable the manipulation of data contained in documents that do not conform to an XML structure [DBMSB].

Interface	Function
Document	Top-level representation of an XML document
Node	Representation of any node in the XML tree
Element	An XML Element
Text	A Textual String

Table 2.3 DOM Interfaces [JVCB]

JavaScript-enabled web browsers have always defined a document object model. A web-browser DOM may state, for example, that the forms in an HTML document are accessible through the *forms[]* array⁵ of the Document object (refer to Section 2.6.1).

```

<html>
<head>
  <title>Sample Document</title>
</head>
<body>
  <h1>An HTML Document</h1>
  <p>This is a<i>simple</i>document.
</body>
</html>

```

Code Listing 2.2 Sample HTML code

HTML documents have a hierarchical structure that is represented in the DOM as a tree structure. The nodes of the tree represent the varying types of content in a document. The tree representation of an HTML document primarily contains nodes representing elements or tags such as *<body>*⁶ and *<p>*⁷ and nodes representing

⁵ Using the *forms[]* array each Form object can be accessed in turn. It has a length property, which is required to see how often a function needs to be looped through.

⁶ It represents the main part of the HTML document where most of the coding is done.

⁷ Denotes the start of a paragraph.

strings of text. A HTML document may also contain nodes representing HTML comments. The following is a simple tree representation of an HTML document:

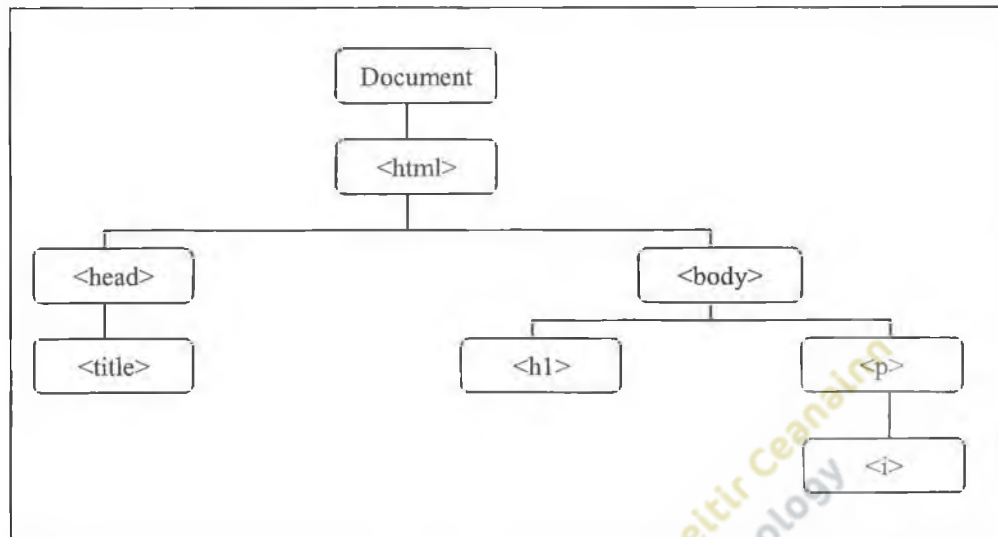


Fig. 2.4 The tree representation of a HTML document

The node directly above a node is the *parent* of that node. The nodes one level directly below another node are the *children* of that node. Nodes at the same level, and with the same parent, are *siblings*. The set of nodes any number of levels below another node are the *descendants* of that node. The parent, grandparent, and all other nodes above a node are the *ancestors* of that node. As a tree-based API, the DOM requires the parser to build a complete tree representation of the document containing nodes that can then be explored [BXAL].

2.6.1 Nodes

The DOM tree structure depicted in Figure 2.4 is represented as a tree of various types of Node objects. The Node interface defines properties and methods for navigating and manipulating the tree [DXDB]. The *childNodes* property of a Node object returns a list of children of the node, and the *firstChild*, *lastChild*, *nextSibling*, *previousSibling*, and *parentNode* properties provide a way to negotiate the tree of

nodes. Methods such as *appendChild()*, *removeChild()*, *replaceChild()*, and *insertBefore()* enable the addition and removal of nodes from the document tree.

The DOM can also be used to represent XML documents, which have a more intricate syntax than HTML documents, and the tree representation of such a document may contain nodes that represent XML entity references, processing instructions, Class Data (CDATA) sections, and so on. A lot of client-side Java programmers do not need to use the DOM with XML documents. JDOM and XDOM are two means of accessing data (apart from DOM and SAX) for Java-programmers. Attributes that carry out 'normal' tasks are accessible via the older object model, like *document.forms[0].item('name').value*.

The DOM standard defines interfaces, not classes as can be seen in Table 2.3 and Figure 2.5. There are two important versions, or "levels", of the DOM standard. DOM Level 1 was standardised in October 1998. It defines the core DOM interfaces, such as Node, Element, Attribute, and Document, and also defines various HTML-specific interfaces. DOM Level 2 was standardised in November 2000 (with DOM Level 3 currently⁸ at a 'working draft' stage). In addition to some updates to the core interfaces, this new version of the DOM is greatly expanded to define standard APIs for working with document events and CSS style sheets and to provide additional tools for working with ranges of documents.

As of Level 2, the DOM standard has been "modularised." The core module, which defines the basic tree structure of a document with the Document, Node, Element, and Text interfaces, is the only required module. All other modules are optional and may or may not be supported, depending on the needs of the implementation as can be

⁸ 2003

seen in Figure 2.5. The DOM implementation of a web browser would support the HTML module, since web documents are written in HTML. Browsers that support CSS style sheets typically support the Style Sheets and CSS modules, because CSS styles play a crucial role in Dynamic HTML programming. CSS allows the programmer to apply styles to HTML and XML documents either from within the document or with a separate style sheet. It can be said that it allows the programmer to separate the concept of content and presentation [BTBF]. Similarly, since most client-side JavaScript programming requires event-handling capabilities, one would expect web browsers to support the Events module of the DOM specification. The Events module was only recently defined by the DOM Level 2 specification⁹.

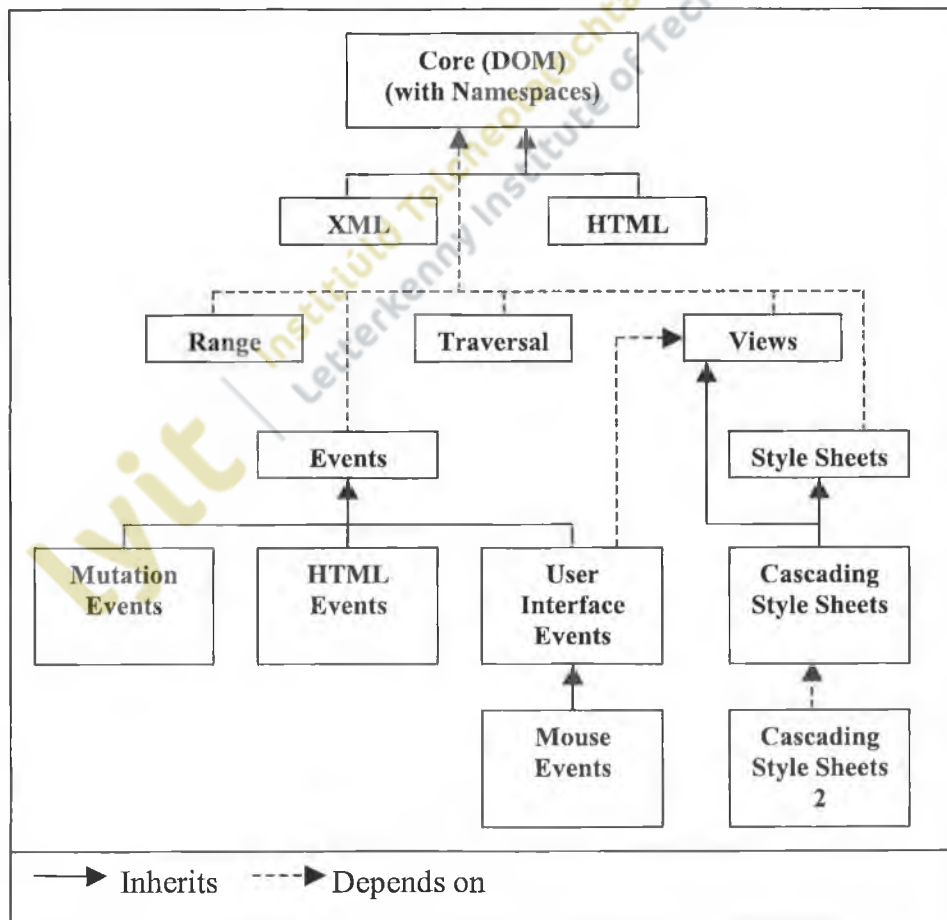


Fig. 2.5 DOM Level 2

⁹ 2003

To summarise, the DOM does not specify a standard way to read XML data into memory. Almost all DOM implementations delegate this task to a dedicated parser and in the case of Java; SAX is the preferred parsing technology [XSLB]. SAX is discussed further in the next section.

2.6.2 SAX

As opposed to being a specification developed by a standards body, SAX was created by members of a mailing list called XML-Dev under the direction of David Megginson [PDBW]. SAX can be regarded as an alternative to the DOM as a technique for accessing XML documents. At the very least it compliments the DOM [PDBW]. Depending on the type of document that is being worked with and the actions that require execution, various advantages exist when using either of the two approaches provided by the DOM and SAX to access and/or manipulate data in XML.

2.6.2.1 SAX Parsers

It is important to realise that SAX is an API, just as the DOM is and can be implemented in a parser [PDBW]. This implies that SAX itself simply provides interfaces and classes to be incorporated by a parser or application. Table 2.4, lists a few examples of some freely available SAX parsers [PDBW]:

Parser Name	Creator	SAX Version	Location (http://)	Language(s) Supported
Aelfred	David Megginson	2.0	home.pacbell.net/david-b/xml#utilities	Java
Saxon	Michael Kay	2.0	users.iclway.co.uk/mhkay/saxon/index.html	Java
MSXML3	Microsoft	2.0	msdn.microsoft.com/downloads.default.asp	C++, VB & all COM-compliant
Xerces C++ Parser	Apache XML	2.0	xml.apache.org/xerces-c/index.html	C++
Xerces Java	Apache	2.0	xml.apache.org/xerces-j/index.html	Java
JAXP	Sun	2.0	java.sun.com/xml/download.html	Java
XP	James Clark	1.0	www.jclark.com/xml/xp/index.html	Java

Table 2.4 Freely available SAX parsers

The most fundamental difference between SAX and the DOM is that SAX is an event driven interface that requires particular methods to be declared that can 'catch' events from the parser [PDBW].

It is important to note that when a DOM-based parser parses an XML document, it stores a tree-based representation of the document in memory. Conversely, SAX streams the document through from start to finish, looking at the various items it encounters as the document passes [PDBW]. For each structural item in the document, SAX calls a method that the programmer has made available as can be seen in Figure 2.6:

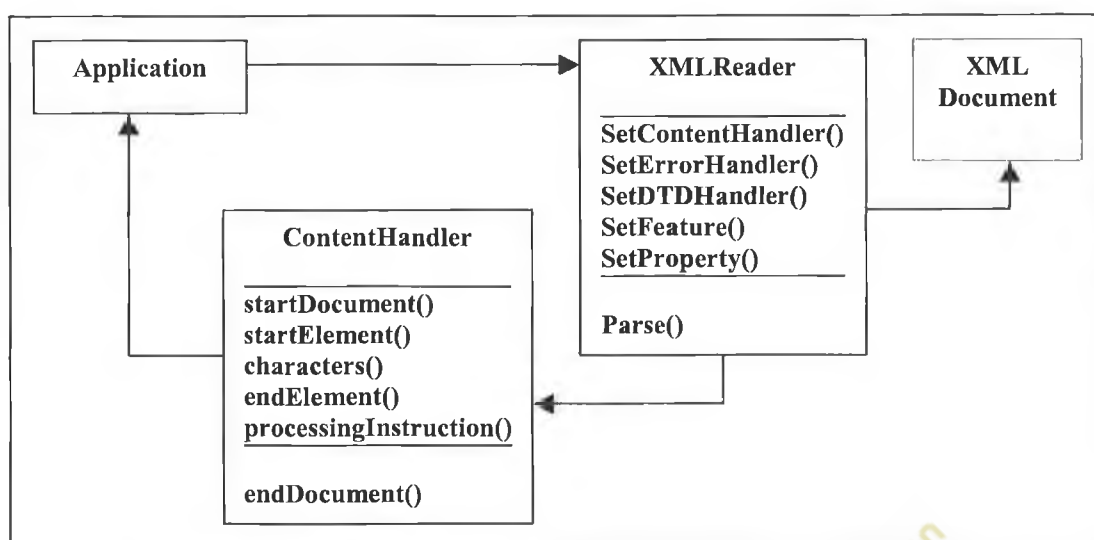


Fig. 2.6 XML Parsing

Figure 2.6 depicts the way in which an XML document can be processed using a filter. A client application instructs the filter, represented in SAX by an *XMLFilter* object, to read the text of an XML document. The filter then instructs the parser to read the text of an XML document. As it reads, the parser calls back to the filter's *ContentHandler*. The filter's *ContentHandler* then calls back to the client application's *ContentHandler*.

2.7 SAX versus DOM

Two of the major types of XML (or SGML) APIs are the Tree-based API (SAX) and the Event-based API (DOM) which are discussed in the next section.

2.7.1 Tree-based APIs – DOM versus Event-based APIs – SAX

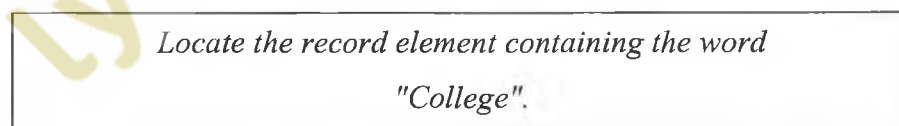
With the DOM a tree-based API maps an XML document into an internal tree structure. The application developed can then traverse its way through that tree

[DOML]. The DOM helps to maintain a tree-based API for XML and HTML documents.

An event-based API provides information about parsing events (such as the start and end of elements) directly to the application through call-backs, and generally does not create an internal tree (e.g. SAX). The SAX API is also available in different programming languages [SAXH]. This can be seen in Table 2.4. The application implements handlers to deal with the various events, similarly to handling events in a GUI.

2.7.2 Incorporating the correct API

Tree-based APIs (such as the DOM) are utilised in a wide range of applications. Nevertheless, they can put a strain on system resources, especially if the document is large [DOML]. An event-based API (such as SAX) can help provide a less complicated, lower-level access to an XML document. It is possible to parse documents much larger than available system memory, and the developer can construct their own data structures using the callback event handlers. For example:



*Locate the record element containing the word
"College".*

Fig. 2.7 Syntax Example for Handling a Callback Event Handler

If an XML document were 25MB in size (or even if it were only 2.5MB), it would be considered to be inefficient to traverse a parse tree just to locate one piece of contextual information. On the other hand, an event-based API would enable a

developer to find the information in a single pass using very little memory. The following example illustrates how an event-based API could work:

<?xml version="1.0"?>	start document
<my_doc>	start element: my_doc
<para>	start element: para
College	characters: College
</para>	end element: para
</my_doc>	end element: my_doc
		end document

Fig. 2.8 Syntax to Illustrate Event-Based API

An event-based interface will break the structure of the document shown in Figure 2.8 down into a series of linear events. An application handles these events just as it would handle events from a GUI. If the information is structured in a way that makes it easy to create an element to object mapping, then the SAX API may be incorporated.

When deciding which method of parsing to incorporate into a system it is best to summarise the various advantages/disadvantages of each [PDBW]:

2.7.3 Advantages of SAX over the DOM

- **Handling large XML Documents:** It is possible for a DOM-parsed document to take up to ten times its own size in Random Access Memory (RAM). Therefore, when dealing with large data sources of 100MB (Mega Bytes) or more, the performance gains with SAX will be more advantageous than those offered by the DOM.
- **Creating your own document as a subset of the whole:** SAX can be advantageous in preparing a document for the DOM or for another parser, by

creating a new smaller document that contains only the pieces that are required.

- Filtering documents in an event stream: SAX provides the '*XMLFilter*' interface for defining classes that interrupt parsing events as they pass from the '*XMLReader*' to a '*ContentHandler*'. By incorporating these features, document transformations could be allowed to occur at several levels.
- Aborting processing: It is possible to escape from SAX processing during any event handler.

2.7.4 Advantages of the DOM over SAX

- Modifying and saving the original document: SAX cannot necessarily modify an XML document and it is considered to be read-only. It is simpler to write to the in-memory document with the DOM.
- Modifying document structure: It is possible to change a document structure in SAX by writing a new document, however, the DOM allows you to make such a change easily.
- Random access; the problem of context: Traversing through an XML document can be problematic, as SAX does not concern itself with all of the details in the entire document. The DOM is a better alternative for complicated document handling.

2.8 Document Type Definition (DTD)

XML documents must begin with an XML declaration while the HTML declaration is optional. Also the HTML declaration does not use the `<.tagname>`¹⁰ (empty tag) notation [DTDG]. If such a tag exists within a document, then it must be considered in order for the document to be considered 'well-formed'. When these rules are applied the document may be considered well formed. The DTD defines valid tags and attributes, and their allowed contents [DTDG]. In the same way as rules manage the writing of XML documents, other rules also manage the writing of XML DTDs. XML's DTD language is essentially a subset of SGML's, therefore XML is seen to be a subset of SGML [DTDG].

The DTD is the grammar for a markup language and is defined by the designer of the markup language. The DTD specifies what elements may exist, what attributes the elements may have, what elements may be established inside other elements, and in what order [DTDG]. The DTD defines the document type and sets out the rules that the XML document must follow [DEAB]. This is the part of XML that creates its Extensibility. When applying a DTD to a document to be sent across the Web, the DTD that is transmitted tells a browser what the tags are called, what they mean and how they can be used [CCMI]. The DTD is the means by which a new markup language is actually defined. DTDs were being written for a large number of different problem domains (e.g. from marking up various pieces of literature to marking up mathematics, etc.), and each DTD defines a new markup language [DTDG]. New markup languages now exist, or are being designed, for example to mark up the plays of Shakespeare. Markup languages are also being developed to model information in

¹⁰ A tag beginning with a full-stop

the health care industry (HL7¹¹ SGML/XML) and languages to typeset, display, and actively use mathematical equations (MathML) [DTDX]. Refer to Section 2.14.

A DTD that describes the tags the markup language utilises, the attributes of the tags, and how they may be combined is vital to each of these new languages. A DTD specifies what information may or may not be incorporated in a markup language. When these rules are adhered to a well-formed XML document is created. Code Listing 2.3 shows a well-formed XML document that adheres to the rules of the DTD shown:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE LIBRARY SYSTEM "Library.dtd">
<library>
  <stock>
    <college>Letterkenny IT</college>
    <book>
      <isbn>00-111-34555</isbn>
      <author_first>Joe</author_first>
      <author_last>Bloggs</author_last>
      <title>Joe's Book</title>
      <rating>****</rating>
    </book>
  </stock>
</library>
```

Code Listing 2.3 A well-formed XML Document called Book.xml

The XML Document shown above in Code Listing 2.3 can only be described as well formed when the rules of the following DTD are applied to it:

¹¹ Health Level 7 – W3C Member

```
<!ELEMENT library (stock)+>
<!ELEMENT stock (college, book*)>
<!ELEMENT college (#PCDATA)>
<!ELEMENT book (isbn,author_first,author_last,title,rating)>
<!ELEMENT isbn (#PCDATA)>
<!ELEMENT author_first (#PCDATA)>
<!ELEMENT author_last (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT rating (#PCDATA)>
```

Code Listing 2.4 A DTD called **Library.dtd**

The tree structure for the code in Code Listing 2.4 might be diagrammatically depicted as follows:

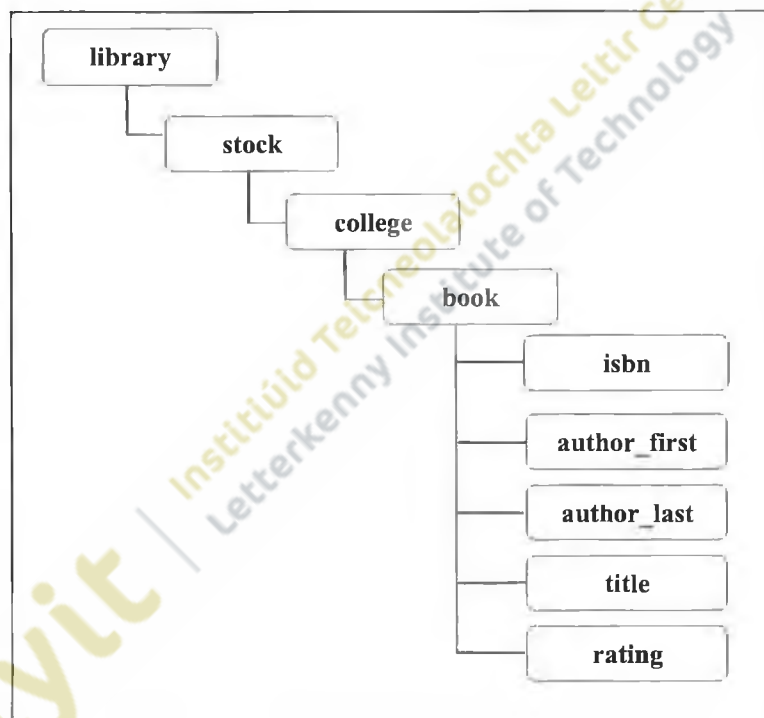


Fig. 2.9 Tree Representation of Code Listing 2.4

The DTD language, which was once the most general means of describing the structure of XML instance documents, may now lack enough expressive power to appropriately describe highly structured data [ACMR]. However, XML Schema (refer to Section 2.8.3) provides a better set of structures as well as types and constraints for

describing data and is consequently expected to become a quite common process for defining and validating highly structured XML documents [ACMR].

2.8.1 Cascading Style Sheet (CSS)

The Cascading Style Sheet (CSS) can be applied to an XML file to format a web page so that the code itself (within the XML) can be less burdened with formatting and more focused on content. It helps transform a plain XML document into a readable and formatted web page. Internet Explorer 5 and later can directly display XML files with or without an associated stylesheet. If no stylesheet is provided, then the browser uses a default built-in XSLT stylesheet (refer to Section 2.19) that displays the tree structure of the XML document.

However, it is possible for a developer to write their own stylesheet saved with the extension '.css'. Code Listing 2.5 shows an example of CSS code that could be applied to an XML document with the associated tags (i.e. *catalogue* and *cd*):

```
catalogue
{
    background-image: url(My_Pic.jpg);
    background-attachment: fixed;
    width: 100%;
    border: inset;
    padding: 70px;
}

cd
{
    list-style-image: url(image.gif);
    display: block;
    margin-bottom: 10pt;
    margin-left: 0;
}
```

Code Listing 2.5 CSS Code

One of the main advantages of the CSS is that it allows a developer to dynamically link a style sheet, and as a result only one file needs to be changed in order to change the appearance of the entire Web site.

2.8.2 XML Namespaces

An XML namespace is a mechanism to identify XML elements [XEGP]. A namespace is a solution to help manage XML extensibility. This is necessary because with distributed environments conflicts could arise as a result of the extensibility not being managed properly. XML is a flexible language and it was created under the assumption that it could be extended by anybody. However this would inevitably lead to problems. In order to best describe the problem and how to address it, it is necessary to apply an example. Assuming somebody has created an XML document that describes video details, i.e. `<name>`, `<price>` and `<rating>` for example. If the original creator were to use the `<rating>` tag to assign 'star' values (e.g. 3 stars) to a video and then someone else extended the file by placing an additional rating tag on to the file, problems would arise. The second person might want to place a parental advisory description to the document (e.g. PG). If a piece of software were used to display this document problems would arise because there would be two `<rating>` tags offering different information. This is where a namespace is required. The namespace qualifies element names uniquely on the Internet in order to avoid conflicts between elements with the same name. The namespace is identified by some Uniform Resource Identifier (URI), either a Uniform Resource Locator (URL), or a Uniform Resource Number (URN), but it does not matter what, if anything, it points to [XEGP]. URIs are used because they are globally unique across the Internet. Code

Listing 2.6 is an example of a namespace that might be declared at the beginning of an XML file:

```
<?xml version="1.0"?>
<references xmlns:pa= http://me.you.com/ref/3.3>
  <name>...</name>
  ...
  <pa:rating>G</pa:rating>
</references>
```

Code Listing 2.6 Applying a Namespace to an XML file

The prefix uniquely identifies the type of rating that this document will incorporate. The prefix alone would not solve the problem which is why it is associated with a URI which is guaranteed to be unique [XEGP].

Namespaces can be declared either explicitly or by default. With an explicit declaration, a shorthand definition is incorporated by the developer, or prefix, to substitute for the full name of the namespace. To relate this to the example `<rating>` would have a prefix (e.g. `<pa-rating>` for parental advisory). This prefix is then used to qualify elements belonging to that namespace. In order to have a valid document the prefixes must also be declared in the DTD. It is the author's opinion, that perhaps the use of 'namespaces' may reduce the ability of XML to filter data as general searches may return data that was not sought.

2.8.3 XML Schemas

A schema is any type of model document that defines the structure of something [BXML]. XML Schemas help to convey the data syntax and semantics for applications such as B2Bs [NMSX]. Benefits of XML Schemas [BXML]:

- Created using XML.
- Fully supports the Namespace Recommendation.
- Allows validation of text element content based on built-in and user-defined data types.
- Allows for an easier creation of complex and reusable content models.
- Allows modelling of concepts such as object inheritance and type substitution.

An XML document that is described by a schema is called an 'instance document' [XMLN]. If a document complies with all the constraints specified by the schema, it is considered to be 'schema-valid'. Code Listing 2.7 shows an example of a valid XML Schema that can be applied to an XML file.

```
<?xml version="1.0"?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data" xmlns:dt="urn:schemas-microsoft-com:datatypes">
  <ElementType name="date" dt:type="dateTime"/>
  <ElementType name="description"/>
  <ElementType name="book"/>
  <ElementType name="isbn"/>
  <ElementType name="Books" content="eltOnly">
    <group minOccurs="0" maxOccurs="1">
      <element type="description"/>
    </group>
    <group minOccurs="0" maxOccurs="1">
      <element type="date"/>
    </group>
    <group minOccurs="1" maxOccurs="1">
      <element type="library"/>
    </group>
  </ElementType>
  <ElementType name="library" content="eltOnly">
    <element type="book"/>
    <element type="isbn"/>
  </ElementType>
</Schema>
```

Code Listing 2.7 XML Schema Example

The `<Schema>` element is the root element within an XML Schema. It facilitates the declaration of namespace information as well as any defaults for declarations throughout the document. The `'dt:type'` indicates the data type of the element whereas the `'content'` is an indicator of whether the content must be empty or can contain text, elements, or both. In the example in Code Listing 2.7 the value is `"eltOnly"`. This means that the element can only contain the specified elements. It cannot contain any free text. Sometimes it is necessary to have a "flow" within a document. This can be accomplished in an XML Schema by incorporating a sequence. First of all, the content must be organised into a `<group>` in order to specify the sequence. In the example in Code Listing 2.7 the attributes `'minOccurs'` and `'maxOccurs'` are given numerical values. Code Listing 2.8 shows an example of an XML file that might incorporate the XML Schema declared above in Code Listing 2.7.

```
<?xml:stylesheet type="text/xsl" href="bookxsl2.xsl"?>
<Books xmlns="x-schema:book-schema2.xml">
  <description>Library Books</description>
  <date>2002-11-08T14:00:00</date>
  <library>
    <book>SVG Essentials</book>
    <isbn>11-098987-6</isbn>
  </library>
</Books>
```

Code Listing 2.8 Applying an XML Schema to an XML File

With all the advantages that XML Schemas have to offer when validating a document it would be wrong to suggest that the DTD has been left with no use. Schemas provide no support¹² for the functionality that 'ENTITY' plays in a DTD. Many XML documents rely heavily on support for this element. A DTD can also be embedded

¹² November 2002

within the XML file itself unlike most other syntaxes, which need a separate file [BXML]. With XML data structures, the metadata is defined separately using either a DTD or an XML Schema. If the XML document does not have a DTD or a Schema then its use is limited as developers cannot be certain if the document is valid or which constraints apply [SWDH].

2.9 XML Data Islands

There is an increasing need to be able to embed "islands" of data inside HTML pages. In Microsoft Internet Explorer 5.0 and later, these data islands can be written in XML. The W3C expects to evolve the HTML specification to include the capability of embedding XML in HTML documents. A data island is an XML document that exists within an HTML page [XTUT]. It allows the developer to script in conjunction with the XML document without having to load it through script or through the `<OBJECT>` tag. Almost anything that can reside in a well-formed XML document can reside inside a data island [XTUT]. The `<XML>` element indicates the beginning of the data island, and its 'ID' attribute provides a name that can be used to reference the data island.

In the example shown in Code Listing 2.9, an XML Data Island with an ID "myid" is loaded from an external XML file. An HTML table is bound to the Data Island with a data source attribute, and finally the tabledata elements are bound to the XML data with a data field attribute inside a span.

```
<html>
<body>
<xml id="myid" src="My_XML_File.xml"></xml>
<table border="1" datasrc="#cdcat">
<tr>
<td><span datafld="AUTHOR"></span></td>
<td><span datafld="BOOK_TITLE"></span></td>
</tr>
</table>
</body>
</html>
```

Code Listing 2.9 Example of an XML Data Island

By incorporating Data Islands into a Web page it would be relatively simple to use an XML file as a means of providing information that can be searched through or viewed in an HTML page using some JavaScript. Such a facility allows for a greater sense of functionality within an application and access to a wider range of information more quickly.

2.10 Chosen Applications

While there are many examples of new languages offered by XML that incorporate Schemas, DTDs, XSLTs, etc., it has been decided that only the more prominent of these languages would be outlined in this document. The chosen applications are:

1. Vector Markup Language (VML) – this was chosen in order to show a comparison between itself and SVG in relation to file size and so forth.
2. Scalable Vector Graphics (SVG) – this was chosen to demonstrate its interaction with the DOM and how it compares to VML.
3. Synchronised Multimedia Integration Language (SMIL) – this was chosen to demonstrate how SMIL can bring multimedia to the Internet in many different ways and what support is available for it.

4. Math Markup Language (MathML) – this was chosen to demonstrate how mathematical notations can now be brought to the web through a language and not just as a GIF or a JPEG. It was also chosen to demonstrate the lack of support for some XML technologies.
5. Speech Markup Language (SpeechML) – this was chosen in order to show a general comparison in code compared with VoiceXML.
6. Voice Markup Language (VoiceXML) – this was chosen to show the capabilities of VoiceXML in comparison to SpeechML. Speech recognition will play an ever-increasing role in the future of the Internet.

2.11 Vector Markup Language

The Vector Markup Language (VML) supports vector graphic information in the same way that HTML supports textual information and has been available since 1998. Within VML the content comprises of paths described using connected lines and curves. The markup gives semantic and presentation information for the paths. VML is a subset of XML and is written using XML syntax. It is possible for Web developers to cut and paste vector graphics from one position to another and edit them with little alteration to the quality of the graphic [RLVM]. For a small VML graphic embedded in a larger document, the DOM is ideal, as it provides an easy way for a scriptwriter to manipulate the graphic or allowing the ability to make minor alterations to a presentation without requiring the use of special software tools.

Vector graphics take up less space in memory than bitmapped/raster graphics for example and as a result display more quickly over slow network connections than traditional GIF and JPEG bitmap images [WPIV]. In vector graphics, the file that

results from a graphic being created is saved as a sequence of vector statements. For example, instead of containing a bit in the file for each bit of a line drawing, a vector graphic file describes a series of points to be connected. As a result, the file size is reduced. VML can support a variety of vector graphics formats/types making it very versatile.

VML is an XML-based exchange, editing, and delivery format for high-quality vector graphics on the Web [WPIV]. An XML-enabled vector-based graphical language can include all kinds of information that might be delivered along with the graphic. Adding metadata to graphics can enhance new applications without significantly adding to the document's file size or download time. With these characteristics VML meets the requirements of both productivity users and graphic design professionals. However, the author noted that reduction of the file size due to the use of VML is dependent on the type of image displayed (e.g. triangle versus a photograph of a person).

VML is unsupported by most web browsers at this time¹³ (refer to Chapter 5, Table 5.2) thus making its capabilities relatively unknown. However, Microsoft Internet Explorer 5 and Microsoft Office currently support VML for Windows 95, Windows 98, and Window NT 4.0. VML code is ignored however by other browsers [WPIV]. Microsoft Word, Microsoft Excel, and Microsoft PowerPoint can be used to create VML graphics. VML has been proposed to the W3C as a standard for vector graphics on the Web. Code Listing 2.10 shows VML code that will display a simple red square to the screen:

¹³ 2002


```

<!--define vml namespace-->
<html xmlns:vml="urn:schemas-microsoft-com:vml">
<head>
  <title>Shapes created using VML code</title>
  <object id="VMLRender" classid="CLSID:10072CEC-8CC1-11D1-
    986E-00A0C955B42E">
</object>
<style>
  vml\:* { behavior: url(#VMLRender) }
</style>
</head>
<body>
  <div>
    <vml:rect style="height: 80px; width=93px" strokecolor="black"
      fillcolor="red">
    </vml:rect>
  </div>
</body>
</html>

```

Code Listing 2.10 VML Code to display a Red Square

The line of code that begins ‘*vml\:**’ instructs the browser to pass all of the tags beginning with ‘*v:*’ to the rendering object. If declaring the behaviour is omitted then the VML will not be displayed.

XML-based technologies are constantly being improved and progress is being made towards enhancing Web based standards. VML facilitates faster graphic downloads. It allows the deliverance of high-quality, fully integrated, scalable vector graphics to the Web, in an open text-based format. Rather than referencing graphics as external files, VML graphics are delivered inline¹⁴ with the HTML page, allowing them to interact with users. Figure 2.10 depicts how a VML structure is created:

¹⁴ Images as part of the page – next to or surrounded by text

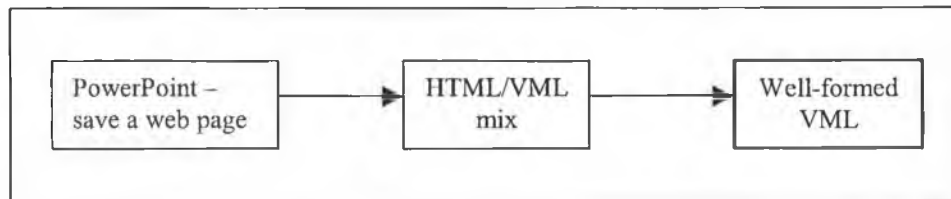


Fig. 2.10 VML Structure

When the slides are saved in PowerPoint 2000, a HTML file is created for each slide, with any graphics that were created being converted to VML [XMLH]. As well as the VML data, the HTML files contains both text and image data. When the slides are created in this way it becomes possible to integrate simple slide presentations using Microsoft Internet Explorer 5, however not for editing and integrating them any further into an XML system [WPIV]. VML applications tend to rely on the CSS therefore a simple task of displaying a rotating caption (captions on X and Y-axis of a graph) can prove quite difficult. This is where the capabilities of Scalable Vector Graphics (SVG) could pose as a more appealing option.

2.11.1 Advantages of VML

- The markup gives semantic and presentation information for the paths.
- It is possible to edit graphics with little alteration to the quality of the graphic.
- Vector graphics take up less space in memory than bitmapped graphics for example and as a result display more quickly over slow network connections than traditional GIF and JPEG bitmap images.
- Adding metadata to graphics can enhance new applications without significantly adding to the document's file size or download time.

2.11.2 Disadvantages of VML

- VML is unsupported by most web browsers thus making its capabilities relatively unknown.
- VML applications tend to rely on the CSS therefore a simple task of displaying a rotating caption (captions on X and Y-axis of a graph) can prove quite difficult.
- The capabilities of SVG could pose as a more appealing option.
- VML graphics do not reduce file sizes for all graphics types.

2.12 Scalable Vector Graphics

Scalable Vector Graphics (SVG)¹⁵ is an innovative graphics file format and Web development language based on XML. The SVG format is emerging through the efforts of the W3C and its members (proposed version is SVG 1.2, W3C Working Draft 29th April 2003). Adobe has taken a leadership role in the SVG specification. SVG allows Web developers and designers to create dynamically generated, choice graphics from real-time data with accurate structural and visual management [KNOG]. It can be defined as a form of XML that can be used for describing two-dimensional graphics [SDPA]. It is possible to create graphics with filter effects, scripting and animation. With this powerful new technology, SVG developers can create Web applications based on data-driven, Graphical User Interface (GUI), and graphics created personally. SVG graphics can be dynamic and interactive. The DOM for SVG allows for simple and efficient vector graphics animation via scripting. An extensive set of event handlers such as *'onmouseover'* and *'onclick'* can be assigned to any SVG graphical object. Because of its compatibility with and leveraging of

¹⁵ SVG 1.1 Specification, W3C Recommendation 14th January 2003.

other Web standards, features like scripting can be carried out on SVG elements and other XML elements from different namespaces simultaneously within the same Web page.

As a result of SVG being written in XML, the data can be linked to back-end business processes such as E-Commerce systems and commercial databases [KNOG]. SVG data can be modified for many different groups of customers. It is possible for Non-Roman and additional atypical fonts and typefaces to be embedded in an SVG document [KNOG]. SVG has the flexibility to make graphics adaptable to all users, regardless of how the users interact with the graphics (via desktop browser, PDA, mobile phone, etc.). Essentially a single file is created that can be easily implemented into various situations. SVG works well with style sheets to manage presentation elements. CSS can be used, not only for font characteristics (size, font-type, and colour), but for properties of other SVG graphic elements as well. For example, it is possible to control and change the stroke¹⁶ colour, fill colour, and clarity of a geometric shape from an external style sheet. Again, the author notes that not all graphics reap the full benefits of file size reduction associated with vector graphics.

Figure 2.11 depicts the layout of an SVG file:

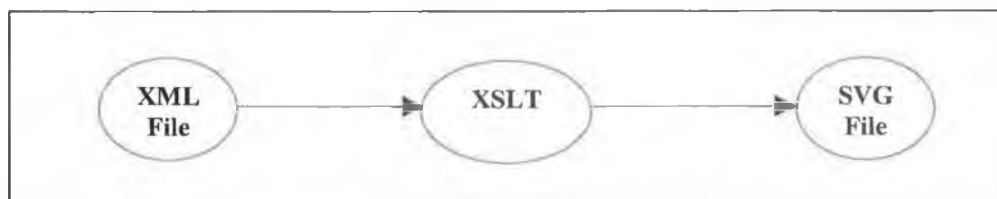


Fig. 2.11 SVG Layout

¹⁶ The stroke colour is used to frame shapes and lines with colour.

SVG is text based therefore coding techniques can be learned relatively quickly. Because SVG can be easily integrated into other applications a Java based toolkit called Batik has been created that uses SVG for formatting functions such as viewing, generating or manipulating graphics [BATK]. Figure 2.12 shows how Batik operates with SVG:

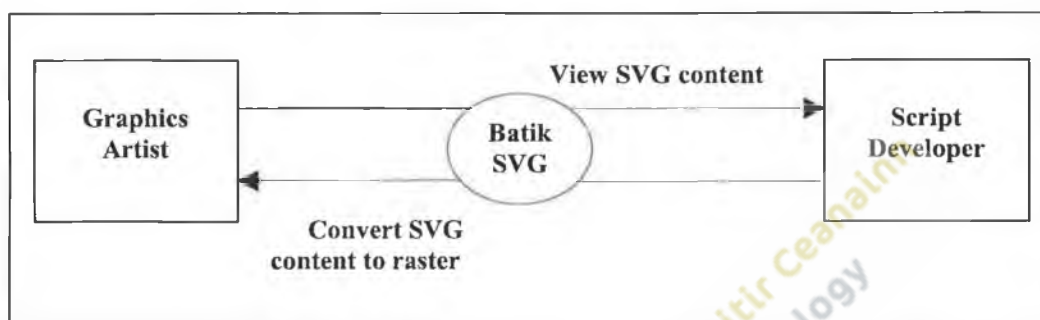


Fig. 2.12 Conversion of SVG using Batik [BATK]

Batik creates an adaptable environment for Java based applications to cope with SVG content. For example, using Batik's SVG generator, a Java application can easily export its graphics in the SVG format. Using Batik's SVG processor and SVG Viewing component, an application can very easily integrate SVG viewing capabilities. [BATK]

With XML, a browser's rendering engine determines the most appropriate way to resolve an image for the host device. SVG images can basically scale up or down and deliver to whatever the receiving device's best solution is [SVG]. However, bitmapped graphics' resolutions are set for a certain image-map size and as a result of this, it distorts when scaled to work with a small display or a high-resolution printer [SVG].

Figure 2.13 depicts an SVG viewer and is based on a number of standardised technologies. Animated-SVG viewers work with the document object model and the DOM CSS. This allows programmers to generate and adapt Web pages as program objects [SVG]. SMIL uses XML tags to synchronise a document's multimedia elements.

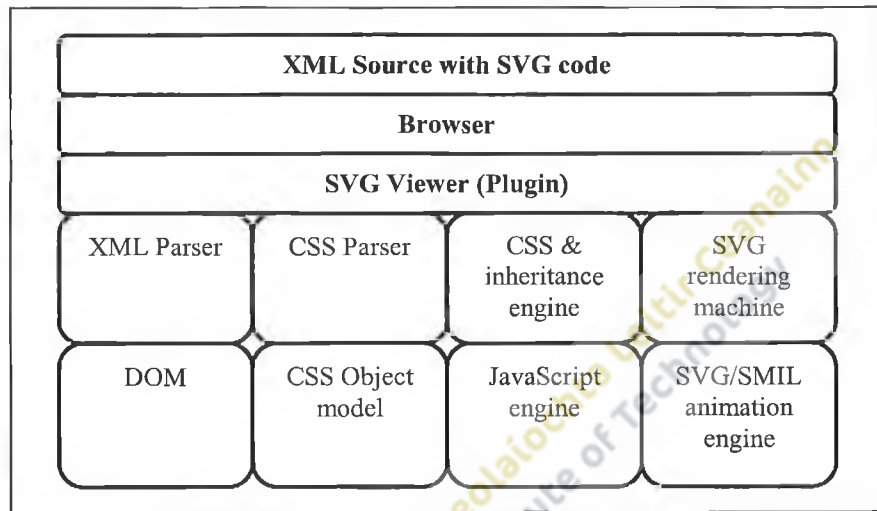


Fig. 2.13 A typical SVG viewer [SVG]

SVG graphics take up less space in memory than GIFs or JPEGs for example; therefore the graphics created in SVG will be displayed more quickly. SVG can be easily integrated into existing technologies and is compatible with the Document Object Model (DOM). This means that enhanced scripting possibilities exist in the form of Geographic Information Systems (GIS) for example. Code Listing 2.11 shows SVG code to create a red square and display it as a graphic when an appropriate SVG Viewer is incorporated such as Adobe SVG Viewer:

```
<?xml version="1.0" encoding="iso-8859-1"?><!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN" "http://www.w3.org/TR/2000/CR-SVG-20001102/DTD/svg-20001102.dtd">
<svg xmlns="http://www.w3.org/2000/svg">
  <rect stroke="black" fill="red" y="51px" x="78px" width="93px" height="80px"/>
</svg>
```

Code Listing 2.11 SVG Code to display a Red Square

It is interesting to note that from the research carried out, the code created in Code Listing 2.10 (see Page 44) to display a red square using VML code created a file 421 bytes in size as opposed to the same geometric shape being created in SVG (Code Listing 11), which resulted in a file 365 bytes in size. In other words, in this example the SVG file was approximately 14% smaller in file size than the VML file.

There are many advantages that SVG can offer with regard to rendering graphics on to Web pages, such as file size reduction, etc., however certain problems still exist. For example, a problem that is prominent is that the capabilities of certain devices are becoming increasingly more varied. Displays are used in all shapes and sizes, ranging from PDAs and mobile phone screens to large conference room displays [ACMM]. Viewers want different things from different applications and so it is becoming increasingly difficult for the creator of these applications to determine what users will require. Therefore a fixed design may not be suitable in various situations [ACMM].

2.12.1 Advantages of SVG

- SVG allows Web developers and designers to create dynamically generated, high-quality graphics from real-time data with very precise structural and visual management.
- As SVG is written in XML, the data can be linked to back-end business processes such as E-Commerce systems, commercial databases, and similar sources of real-time information.

- SVG is text based therefore coding techniques can be learned relatively quickly.
- SVG graphics take up less space in memory than GIFs or JPEGs for example.
- Interoperability. SVG can interact with other languages (e.g. JavaScript).
- Internationalisation (Unicode support).
- Wide tool support.
- Easy manipulation through standard APIs, such as the DOM API.
- Easy transformation through XML Stylesheet Language Transformation (XSLT).

2.12.2 Disadvantages of SVG

- SVG code will only display a graphic when an appropriate SVG Viewer is incorporated such as Adobe SVG Viewer.
- Support for SVG is limited.
- Resolutions may vary, and certain devices, such as mobile phones, do not have full colour support as of yet.
- Viewers want different things from different applications and so it is becoming increasingly impossible for the creator of these applications to determine what users will require.
- A fixed design may not be suitable in certain situations.
- SVG does not reduce file sizes for all graphics types.

2.12.3 Graphics Comparison

To conclude, there are primarily two types of computer graphics: bitmapped and vector. A bitmapped graphic contains a list of colours of individual pixels in a

normally rectangular area [XMLH]. Examples include GIF, JPEG and Portable Network Graphics (PNG) images used on most Web pages. If a bitmapped graphics is 5 inches by 5 inches and has a resolution of 72 pixels per inch, then it contains 72 x 5 x 72 x 5 pixels, that is, 129,600 pixels. If the image is stored in 24-bit colour, then each pixel occupies 3 bytes, so this image uses 3,110,400 bits (which is approximately 379.6KB¹⁷ of memory). It is possible for the actual file to incorporate a variety of lossy and nonlossy compression algorithms to reduce the size. However, the calculations above shows how bitmapped images can increase in size very quickly. As a result of these large files, many Web pages with multiple graphics are slow to render images. By comparison, a vector graphic does not store several bytes of data for each pixel in the image. Instead it stores a list of instructions for drawing the image. Generally the space required for a vector graphic is much less than what is required for a bitmapped image [XMLH]. For this reason it can be concluded that SVG is helping to reduce bandwidth requirements and improve download times on the Internet.

2.13 Synchronised Multimedia Integration Language

Synchronised Multimedia Integration Language (SMIL)¹⁸ is a markup language designed to be easy to learn and install on Web sites. SMIL was created specifically to solve the problems of coordinating the display of multimedia on Web sites. By using a single time line for all of the media on a page their display can be properly time coordinated and synchronised. SMIL is used to describe the behaviour and layout of multimedia presentations giving them some semantic meaning [MRKM]. SMIL also

¹⁷ Calculation carried out in order to determine the Kilobytes of memory in use by the graphic:
(3110400/1024)/8

¹⁸ SMIL 2.0, W3C Recommendation 7th August 2001.

helps to eliminate the necessity to embed low bandwidth data such as text in high bandwidth data such as video just to combine them [XMLH]. SMIL is a tool used for building synchronous, streaming multimedia presentations that integrate audio, video, images, and text. The most popular SMIL browser is the Real Audio G2 player. SMIL applications can be written in much the same way as VML applications however, there is also an editor available called RealPresenter, that allows the synchronisation of video and audio tracks with a PowerPoint presentation.

In a SMIL presentation, all of the media elements including images, audio clips, video clips, animations, and formatted text are referenced from the SMIL file, which is comparable to the way a HTML page references its images, applets, and other elements. The first commercial SMIL player to appear on the market was RealNetworks' "RealPlayer G2". While earlier versions of RealPlayer played only RealNetworks' audio and video file formats, G2 includes support for many other media types such as WAV, AVI, JPEG, MPEG, and others as can be seen in Table 2.5. Roxia Co. has recently (May 2003) released a SMIL 2.0 player and authoring tool called RubiC.

Media	Tag	G2	GriNS	Soja
GIF	Img	Yes	Yes	Yes
JPEG	Img	Yes	Yes	Yes
MS Wav	Audio	Yes	Yes	No
Sun Audio	Audio	Yes	Yes	Yes
Sun Audio Zipped	Audio	No	No	Yes
MP3	Audio	Yes	No	No
Plain Text	Text	Yes	Yes	Yes
Real Text	Textstream	Yes	No	No
Real Movie	Video	Yes	No	No
AVI	Video	Yes	Yes	No
MPEG	Video	Yes	Yes	No
MOV	Video	Yes	No	No

Table 2.5 Media support in SMIL [MDMM]

G2 also supports a number of custom XML-based data types that provide additional features for animating text and images and providing interactivity. Figure 2.14 depicts the various elements of a SMIL presentation:

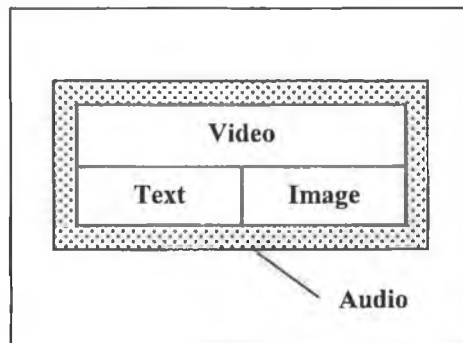


Fig. 2.14 SMIL Elements within a Presentation

The code associated with Figure 2.14 above might look as shown in Code Listing 2.12:

```
<smil>
<head><layout>
<root-layout width="431" height="334" background-color="#000066" />
<region id="pic_pos" width="110" height="48" left="26" top="14" />
<region id="text_pos" width="220" height="48" left="126" top="14" />
<region id="vid_pos" width="110" height="48" left="26" top="62" />
</layout></head>
<body>

<video src="video1.rm" region="vid_pos" begin="1.2s" fit="slice" />
<text src="text1.rt" region="text_pos" begin="1.2s" fit="slice" />
</body>
</smil>
```

Code Listing 2.12 Displaying various media elements using SMIL

Audio, video and animation are presented over a period of time, therefore synchronisation between the sequences is very important. It should be possible to schedule audio, video and animation sequences to take place in a sequence. The difficulties that are evident between these elements can be described in three ways:

time-based, object-based or a combination of both. For example it is important that when a person speaks their lip movements relate exactly to the sound being generated. Poor synchronisation is particularly evident if the sound arrives before the vision.

SMIL includes enhanced functionality for a variety of multimedia services for customers with varying needs. Such diversity can range from desktops to mobile phones, portable disk players, car navigation systems and television sets. Each of these platforms has its particular capabilities and requirements. It is clear that not all of the SMIL 2.0 features are required on all platforms.

A SMIL outline allows a SMIL user to implement only the subset of the SMIL 2.0 standard it needs, while maintaining document interoperability between device profiles built for varying needs. SMIL 2.0 provides a framework for defining a group of scalable profiles. A scalable profile enables users of varying programming abilities with varying technologies to create SMIL documents that will act intelligently and which are tailored to the capabilities of the target devices. Conformance to a SMIL Basic profile provides a basis for interoperability guarantees [XMLU].

Some QuickTime SMIL browsers cannot decode certain media objects thus network problems can arise. Synchronising audio and visual elements can be problematic. This drawback was researched and possible solutions were analysed which can be seen in Chapters 4 (Section 4.2.3), 5 (Section 5.4.2) and 7 (Section 7.7). SMIL 2.0 is the latest version of SMIL and is described as a set of markup modules, which define the semantics, and XML, syntax for certain areas of SMIL functionality [XMLU]. Semantic information is an important factor in XML-based applications. The metadata implies that these applications may prove more useful to other technologies. SMIL could enable tutorials to be streamed over the Internet in various different

educational sectors. With the advancement of XML technologies, Mathematics can now be taught over the Internet. Previously, mathematical notations and symbols proved problematic to display, but XML's Mathematical Markup Language (MathML) has helped to alleviate that problem.

2.13.1 Advantages of SMIL

- SMIL is used to describe the behaviour and layout of multimedia presentation giving them some semantic meaning.
- SMIL includes enhanced functionality for a variety of multimedia services for customers with varying needs.

2.13.2 Disadvantages of SMIL

- Some QuickTime SMIL browsers cannot decode certain media objects thus network problems (such as stalling/faltering) can arise.
- Synchronising audio and visual elements can be problematic.
- Support for SMIL is still quite limited.

2.14 Mathematical Markup Language

Even though the mark-up language HTML has a large list of tags, it does not cater for extensive mathematical displays. With no means of using HTML tags to mark up mathematical expressions, authors have incorporated other means. For example, one popular method involved inserting images. These images were literally snap shots of equations taken from other packages and saved in GIF format and put into technical documents that have a mathematical or scientific content. The W3C had been working with a number of companies with experience in editing and processing mathematics

on computers, as well as other specialist organisations. The work resulted in a markup language called MathML, and the W3C released MathML 1.0 as a Recommendation in April 1998 and MathML 2.0 as a Recommendation on 21st February 2001.

MathML is an XML application that provides a low-level method of communicating mathematics between machines [MMLW]. MathML enables mathematical expressions to be displayed, manipulated and shared over the Internet [MMLD]. An objective of MathML is to enable mathematics to be served, received, and processed on the Web, just as HTML has enabled this functionality for text. A computer algebraic system can encode and evaluate a MathML expression, rendered in a Web browser, edited in a word processor, and printed on a laser printer. Due to this fact MathML is beginning to play an increasingly important role in scientific communications. With MathML it is possible to copy and paste mathematical expressions to and from other places, keeping the integrity of the expression. People with visual disabilities would also be able to read and use mathematical data since it can be interpreted into alternative media such as speech or Braille.

MathML is progressing but is still hindered due to the lack of support for it by browsers [XCMI]. A table showing this support (or lack thereof) can be seen in Table 2.6. However several industries have now taken the necessary steps to support MathML, such as, IBM, makers of the TechExplorer Scientific Browser [MMLD].

		<u>MathML Renderer</u>						
OS	Browser	XSLT	Native	MathPlayer	Techexplorer	Techexplorer	CSS/Java	
			Presentation	Behaviour	Behaviour	Plugin	Script	
Windows	IE X,X<5.0	NO	NO	NO	NO	NO	NO	
	IE 5.0	wd-XSL	NO	NO	NO	YES	NO	
	IE 5.5	wd-XSL	NO	YES	YES	YES	NO	
	IE 5.0/MSXML3	YES	NO	NO	NO	YES	NO	
	IE 5.5/MSXML3	YES	NO	YES	YES	YES	YES	
	IE 6	YES	NO	YES	YES	YES	YES	
	Netscape x,x<=6.0	NO	NO	NO	NO	NO	NO	
	Netscape 6.1	YES	NO	NO	NO	YES	NO	
	Netscape 7.0	YES	YES	NO	NO	NO	NO	
	Mozilla 0.9.4	YES	NO	NO	NO	NO	NO	
	Mozilla 0.9.4/MathML	YES	YES	NO	NO	NO	NO	
	Mozilla 0.9.5+	YES	YES	NO	NO	NO	NO	
	Amaya	NO	YES	NO	NO	NO	NO	
	Opera	NO	NO	NO	NO	NO	NO	
	Lynx	NO	N	NO	NO	NO	NO	
Macintosh	IE 5.0	wd-XSL	NO	NO	NO	YES	NO	
	Netscape x,x<=6.0	NO	NO	NO	NO	NO	NO	
	Netscape 6.1	YES	NO	NO	NO	YES	NO	
	Mozilla	YES	NO	NO	NO	NO	NO	
	Opera	NO	NO	NO	NO	NO	NO	
	Lynx	NO	NO	NO	NO	NO	NO	
Linux/Unix	Netscape x,x<=6.0	NO	NO	NO	NO	NO	NO	
	Netscape 6.1	YES	NO	NO	NO	YES	NO	
	Netscape 7.0	YES	YES	NO	NO	NO	NO	
	Mozilla 0.9.4	YES	NO	NO	NO	NO	NO	
	Mozilla 0.9.4/MathML	YES	YES	NO	NO	NO	NO	
	Mozilla 0.9.5+	YES	YES	NO	NO	NO	NO	
	Amaya	NO	YES	NO	NO	NO	NO	
	Opera	NO	NO	NO	NO	NO	NO	
	Lynx	NO	NO	NO	NO	NO	NO	

Table 2.6 Support for MathML [MMLO]

MathML is an exceptionally extensible mark-up language, which can be used for screen rendering, basic interchange of data between mathematical software applications and generating high-quality print output.

MathML can be used to encode both mathematical notation and mathematical content. As with all XML applications, tags are used to write the application. The tags can be used to Mark-Up a mathematical equation in terms of its presentation and semantics. The main concern of MathML is to present the meaning of equations rather than the graphical presentation of the equation on the user interface. Twenty-eight of the MathML tags describe unusual notational structures, while another seventy-five provide a way of outlining the semantics of an expression.

```

<apply>
  <forall/>
  <bvar><ci> p </ci></bvar>
  <bvar><ci> q </ci></bvar>
  <condition>
    <apply><and/>
      <apply><in/><ci> p </ci><rational/></apply>
      <apply><in/><ci> q </ci><rational/></apply>
      <apply><lt/><ci> p </ci><ci> q </ci></apply>
    </apply>
  </condition>
  <apply><lt/>
    <ci> p </ci><apply><power/><ci> q </ci>
      <cn> 2 </cn>
    </apply>
  </apply>
</apply>

```

Code Listing 2.13 MathML Markup

For example the markup depicted in Code Listing 2.13 would have the notation:

$$\forall p, q \in \mathbb{Q} \wedge p \in \mathbb{Q} \wedge q \in \mathbb{Q} \wedge p < q, \text{ then } p < q^2$$

The presentational tags generally start with "m" and then use "o" for operator, "i" for identifier and "n" for numbers. In Code Listing 2.13,

- `<it>` represents `<`, `<condition>` represents \mathbb{Q} ,

- `<bvar>` represents $|$, `<rational>` represents \mathcal{O} ,
- `<forall>` represents \forall , `<and>` represents \wedge and
- `<power>` represents raising a letter to a power. In this example (Code Listing 2.13), this is the number 2.

The image-based methods that were previously incorporated as the principal means of sending scientific notation over the Web can be considered to be primitive and inefficient.

The following are some of the problems that previously existed when trying to display mathematics on the Web:

- Document quality is reduced
- Authoring is complicated
- Loading can be slow
- Mathematical information contained in images is not obtainable for searching, indexing, or reuse in further applications

MathML is an effective way to include mathematical expressions in web documents. Before MathML became available, the common practice among scientists was to write papers in an encoded format based on the ASCII¹⁹ character set as well as in image format, and e-mail them to each other. In the MathML 1.0 specification there are factors that have not yet been fully developed [MMLW]. It is possible to consider that when macros are introduced into MathML, then programming in it could be less complicated. One possible macro could be an abbreviation macro so that the programmer who coded MathML personally would not have to repeat some

¹⁹ American Standard Code for Information Interchange

complicated but constant notation. There are many different possibilities in MathML, but incorporating a web-based spreadsheet, without the use of applets and plug-ins, may take some time to develop [MMLB]. The main objective of MathML is to present the meaning of equations rather than the presentation of the equation on the screen. MathML may be more difficult to understand than other XML applications therefore, packages are available to help read and write these applications, e.g. Maple [AXML]. In 1999 no web browser supported this innovative technology however since then IBM released a plugin that is suitable for both Netscape and Internet Explorer called the IBM Techexplorer Plugin. This allows MathML code to be transformed into equations rendered to the screen. These markup languages are constantly growing in their notoriety and from the research carried out to date it is becoming increasingly apparent that organisations are seeing the potential for these application and as a result are striving to incorporate them. However, research carried out for this project has identified a few interesting facts about MathML. The following code sample (Code Listing 2.14) was written in Extensible Hypertext Markup Language (XHTML), which eliminated the need for the IBM Techexplorer plugin. It very simply displays the notation $(a+b)^2$:

```

<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="pmathml.xsl"?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head> <meta http-equiv="Content-Type" content="text/html" /> </head>
<BODY BGCOLOR="#1699DA"> <h3 align="center">Math Presentation</h3>
<p><math xmlns="http://www.w3.org/1998/Math/MathML">
  <msup>
    <mfenced open="(" close=")">
      <mrow>
        <mi>a</mi>
        <mo>+</mo>
        <mi>b</mi>
      </mrow>
    </mfenced>
    <mn>2</mn>
  </msup>
</math></p>
</BODY>
</html>

```

Code Listing 2.14 Displaying MathML using XHTML

Code Listing 2.14 incorporates a 'pmathml.xsl' file which is an XSL file that is freely available for download from the W3 web site. It transformed the MathML code embedded within the document into an equation that could be rendered to the screen within the browser. The above code was saved with the .xml extension and was 591 bytes in size. The notation was rendered without any difficulty in Internet Explorer 5.5 but was not rendered properly in Netscape 6.2. Nonetheless, a file was created that did render the given notation in Netscape. It was saved with the .html extension and was 365 bytes in size. Code Listing 2.15 shows its code:

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head><title>Created in 2002</title></head>
<BODY bgcolor="#1699DA"> <h3 align="center">Math Presentation</h3>
<p>
<embed type="text/ezmath"
  pluginspage="http://www.w3.org/People/Raggett/EzMath"
  align="absmiddle"
  alt="(a+b)^2">
</p>
</BODY>
</html>

```

Code Listing 2.15 Displaying MathML in Netscape 6.2

The HTML code written above incorporates the Mathematica language and is embedded within a plugin page. From the research carried out, it could be argued that Mathematica is a better solution to the problem of rendering mathematical notations on the Internet. However, Mathematica is an enormous system and a standard edition takes about 150 MB of memory to store all the relevant files [MMLC]. It is the author's opinion that this defeats the purpose of rendering mathematics using as little of the computer's memory space as possible. From the two examples shown above, the HTML file incorporating the Mathematica notation is smaller in file size. Nevertheless, without XML and the onslaught of MathML, mathematicians would still be taking snapshots of equations created using other packages and embedding those GIFs or Bitmaps into their HTML pages. These equations would simply be images taking up huge amounts of space and users could not interact with the equation nor would any semantic information about the equation be available. A MathML file created using Amaya, for example, can be saved with the extension .mml and rendered in Internet Explorer 5.5 provided the IBM Techexplorer plugin is installed. A .mml file can have the added functionality of providing users with a means of interacting with the equations by clicking on certain numbers/operators, etc. It is the author's opinion that without the creation of MathML, such added functionalities would not have been available (the W3C are currently reviewing a Working Draft of MathML 2.0 (2nd Edition), 11th April 2003). MINSE, which was created in 1996, was one of the first languages employed to help render mathematical notations to the Internet [MMLC]. Its syntax for displaying the notation described earlier would be the same as the Mathematica notation depicted in Code Listing 2.14.

2.14.1 Advantages of MathML

- MathML enables mathematical expressions to be displayed, manipulated and shared over the Internet.
- Enables mathematics to be served, received, and processed on the Web.
- MathML can be used to encode both mathematical notation and mathematical content.

2.14.2 Disadvantages of MathML

- Hindered due to the lack of support for it by browsers.
- In the MathML 1.0 specification there are factors that have not yet been fully developed.
- MathML may be more difficult to understand than other XML applications.

2.15 Speech Markup Language

Speech Markup Language (SpeechML) provides a structure in which web-based applications can incorporate interactive speech capabilities and was introduced in February 1999 by IBM's AlphaWorks [AXML]. SpeechML provides tags for defining spoken output and input as well as tags for initialising actions to be taken on a given spoken input. SpeechML elements are recognised and linked by Uniform Resource Locators (URL) in an effort to keep everything recognisable to HTML developers. Similar to the way HTML can be used as a markup language for creating visual Web applications, SpeechML can be used as a markup language for creating network-based conversational applications [SMLC]. A conversational application is an application that interacts with the user through spoken input and output. A

network-based application refers to one in which the elements of the conversation, that define spoken output and input in SpeechML documents, may be obtained over the network. SpeechML is based on XML, which is a specification for formatting data on Web pages. By using tags a Web site developer can add a conversational interface without having to gain proficiency in speech technology.

SpeechML provides numerous significant advantages over traditional markup languages, including the ability to customise interactions with the user by voice. SpeechML enables the use of a conversational browser to understand streams of markup language data originating from multiple sources [SMLC]. For example, while purchasing an airline ticket, a user can temporarily suspend the transaction and interact with a banking application on a different server to check an account balance. SpeechML like other XML-based markup languages, strives to make complicated technologies functional by normal Web authors through simple and commonly recognised tags. SpeechML can parse data from multiple Web sources so that users can go back and forth between speech-enabled and conventional interfaces. One of the obstacles is the processing power that speech recognition requires, although increasing Central Processing Unit (CPU) speed is helping to alleviate the problem. Another problem with speech recognition technologies is that it would not be appropriate when working with confidential data in an environment where one could be overheard.

As with XML, SpeechML has a set of tags that it implements in order to write the applications. SpeechML is based on XML, which is a specification for formatting data on Web pages. An example of SpeechML can be seen in Code Listing 2.16, which shows a simple example of a menu system:

```
<page>
<menu>
Please choose from the File menu.
<choice target="new">New.</choice>
<choice target="close">Close.</choice>
<choice target="quit">Quit.</choice>
</menu>
</page>
```

Code Listing 2.16 SpeechML Code

SpeechML has many capabilities but VoiceXML is leading the way for voice applications. The capabilities and limitations of SpeechML are outlined below:

2.15.1 Advantages of SpeechML

- SpeechML can be used as a markup language for creating network-based conversational applications.
- Has the ability to customise interactions with the user by voice.
- SpeechML can parse data from multiple Web sources so that users can go back and forth between speech-enabled and conventional interfaces.

2.15.2 Disadvantages of SpeechML

- The processing power that speech recognition requires is quite large.
- SpeechML would not be appropriate when working with confidential data in an environment where one could be overheard.

2.16 Voice Extensible Markup Language

Motorola's VoiceXML, which is similar to HTML, is a Web-based markup language for representing voice applications [VXML]. The VoiceXML language is based on the

XML standard. The VoiceXML 1.0 specification²⁰ was released in March 2000 [PTCI]. However, while HTML assumes a graphical web browser, with display, keyboard, and mouse, VoiceXML assumes a voice browser with audio output, and audio input. VoiceXML utilises the Internet for expanding voice applications and delivering them, which greatly simplifies these complex tasks thus creating new opportunities. The telephone has been very important in the development of VoiceXML. A typical VoiceXML voice browser runs on a specialised voice gateway node that is connected both to the public switched telephone network and to the Internet as can be seen in Figure 2.10. With the possibility of having to incorporate two voice lines an increase in charges could be incurred. These voice gateways extend the power of the web to many of the telephones in the world. VoiceXML is being used by The Weather Channel and CBS Marketwatch.com to deliver audible content and also to interpret spoken input [SMLW].

VoiceXML enables relatively natural dialogues by allowing words to be given a certain articulation outside of `<form>`. For example, certain names may have a completely different pronunciation from what would be imagine judging by the spelling. If a user were to be recognised by their name, “Eilish” and the program was to respond to the user by welcoming them using their name the pronunciation might be incorrect. However, it is possible to specify in the grammar how this name is to be pronounced. Code Listing 2.17 shows how a word can have its pronunciation altered inside tags declared outside of the `<form>` tag. The `<form>` is VoiceXML’s basic dialogue unit, which describes a set of inputs (`<fields>`) required from the user to carry out a transaction between the user agent (browser) and a server [ACML].

²⁰ This is an essential technical requirement for items, materials or services.


```

<ibmlexicon>
  <word spelling="Eilish" pronunciation="e&#618;&#618;l.&#616;&#643;"/>
</ibmlexicon>

```

Code Listing 2.17 Altering the pronunciation of a word in VoiceXML

To better understand how the VoiceXML document interacts with Web Servers and telephones it is necessary to depict this interaction diagrammatically as can be seen in

Figure 2.15:

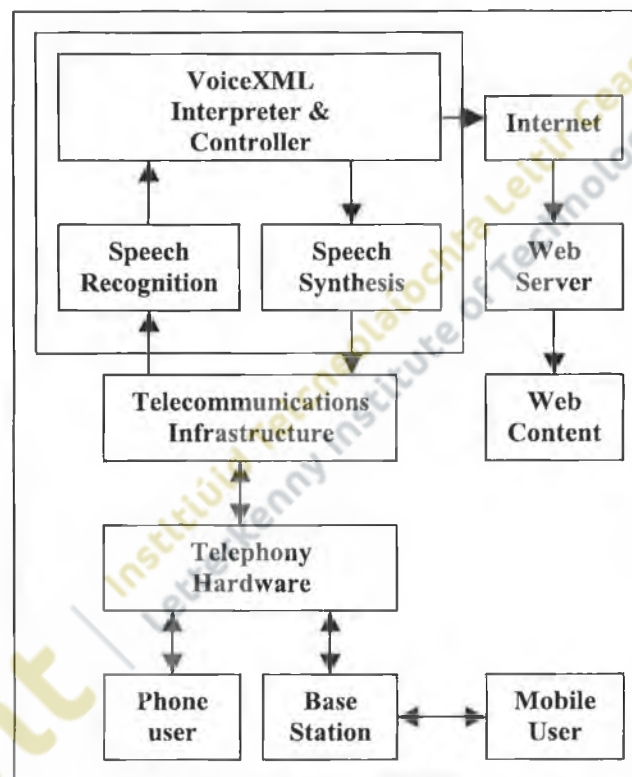


Fig. 2.15 Telephone Interaction with a VoiceXML Interpreter [VDIH]

The user interacts with the system by relating a message via a telephone for example. The message is then recognised by the speech recognition and is then sent to the Voice Interpreter. If the grammar is recognised, then a synthesised response is given to the user or alternatively sent through the Internet to be delivered as Web content in

a browser. Code Listing 2.18 is an example of VoiceXML code that requires the user to give a response when prompted.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<vxml version="1.0">
<form>
<record name="recording">
<prompt>
<audio src="ListenCarefully.au" />
Please leave your message and then press any
key on the keypad, or say "end of recording", to stop.
</prompt>
<grammar>end of recording</grammar>
</record>
<filled><prompt>
You said
<value expr="recording" /
</prompt></filled>
</form>
</vxml>

```

Code Listing 2.18 VoiceXML Code

The top-level element is `<vxml>`, which is mainly a container for dialogues. There are two types of dialogues: forms and menus. Forms present information and gather input whereas menus offer choices of what to do next. In Code Listing 2.18 an audio file “*ListenCarefully.au*” is played to the user asking them to listen to the instructions carefully. Audio is played from a VoiceXML page by referencing the file using an `<audio>` tag, in much the same way as placing a reference to a streaming audio file in a web page.

The program then prompts the user to “*Please leave your message...*”. The prompt element controls the output of synthesised speech and prerecorded audio. After the prompt has stopped the program will record everything that the user says until they have said the phrase, “*end of recording*” or have pressed a key on the keypad. This phrase is a pre-defined grammar, which the program recognises in much the same

way as a Java program would recognise a variable assigned a value. The recording is stored in the field item variable, which can be played back or submitted to a server. The program will then conclude by playing back what the user input. The program as given in Code Listing 2.18 has been written and tested using the IBM WebSphere Voice Toolkit and was found to be 438 bytes in size.

As with all XML application interoperability is an objective and VoiceXML is no exception. VoiceXML protects application developers from difficulties such as a set of processes running at the same time and platform specific APIs [VCMI]. The protection attained provides greater service portability between platforms. Several organisations are taking advantage of this interoperability. Along with Motorola's VoiceXML, AT&T has its version of VoiceXML called Voice Extensible Markup Language (VXML). VXML technology allows a user to interact with the Internet through voice-recognition technology. Instead of a conventional browser that depends on a combination of HTML and keyboard and mouse, VXML relies on a voice browser or the telephone [AXML]. When using VXML, the user interacts with a voice browser by listening to audio output that can be either pre-recorded or computer-synthesised and submitting audio input through the user's natural speaking voice or through a keypad, such as a telephone. VoiceXML technology allows the application interface to be in the form of dialogues. Input is produced through speech recognition of a user's voice and output is produced through text-to-speech (TTS) technology.

Each VoiceXML application has a document, which specifies every interaction dialogue that a VoiceXML interpreter manages [SCMI]. The input provided by the user can affect dialogue interpretation, therefore, the system gathers the data into

requests that it submits to a document server. This document server can then reply with another VoiceXML document so as the user can continue their sessions with other dialogues [SCMI]. Dialogues play an important role in VoiceXML applications. Features of the dialogue include: the compilation of input, creation of audio output, handling of events running at different times and continuation of the dialogue [VCMI]. VoiceXML supports the following input forms:

- Audio recording,
- Automatic speech recognition and
- Touch-tone.

Aside from, Motorola's VoiceXML; AT&T, IBM, Lucent Technologies, and Motorola joined to develop the VXML Forum in a joint effort to endorse the emerging technology. The Forum produced VXML 1.0 as the initial version of VXML. VXML technology allows a user to interact with the Internet through voice-recognition technology.

VXML would be relatively easy to learn for experienced Web programmers as it has an XML-based definition with an HTML-like appearance. Tools that support desktop development of VXML Web applications could also easily process it. In order to fully understand how these languages operate it is necessary to depict diagrammatically how they interact in a given system. Figure 2.16 shows the voice markup languages from a systems-level perspective:

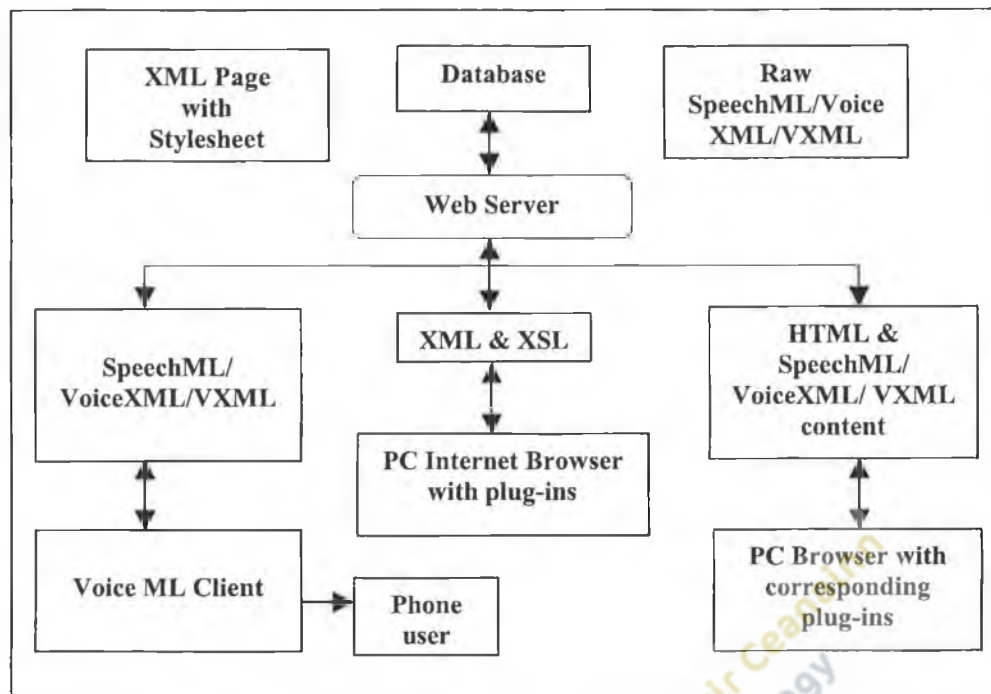


Fig. 2.16 Markup languages at a systems-level [AXML].

At present²¹ it is quite difficult to embed a VoiceXML file into a HTML document. However a new specification has been released in July 2002 called Speech Application Language Tags 1.0 (SALT) [SALTC]. SALT is comprised of a small set of XML elements, with associated attributes and DOM object properties, events and methods, which may be used in conjunction with a source markup document to apply a speech interface to the source page [SALTC]. The SALT specification incorporates a lightweight approach in order to enable speech applications [XJVM]. There are two main differences: VoiceXML has a built-in control flow algorithm, SALT does not and SALT defines a smaller set of elements compared to VoiceXML [XJVM]. The four main tags available with SALT are: *<prompt>* tags (for speech output), a *<listen>* tag (for speech input), a *<DTMF>* tag (for telephone touch-tone input) and a *<smex>* tag (for communicating with other systems) [LNVS]. It is possible to create speech enabled applications using two languages but perhaps the development of one

²¹ October 2002

such language would be more advantageous to the industry. It is the author's opinion that VoiceXML²² will continue to advance, utilising more sophisticated features in support of natural dialogue.

2.16.1 Advantages of VoiceXML

- VoiceXML enables relatively natural dialogues by allowing words to be given a certain articulation outside of *<form>*.
- VoiceXML protects application developers from difficulties such as a set of processes running at the same time and platform specific APIs.
- As it has a HTML-like appearance, VoiceXML would be relatively easy to learn for experienced Web programmers and would be easily processed by tools that support desktop development of VoiceXML Web applications.

2.16.2 Disadvantages of VoiceXML

- At present²³ it is quite difficult to embed a VoiceXML file into a HTML document.
- SALT defines a smaller set of elements compared to VoiceXML, which could possibly help reduce file size.

2.16.3 Summary

VoiceXML provides a larger set of XML elements, since it is intended as a complete, standalone markup. Hence, VoiceXML includes tags for data (forms and fields) and control flow. An execution environment made available to VoiceXML controls the

²² VoiceXML 2.0, W3C Candidate Recommendation 20th February 2003.

²³ October 2002

speech input and output. VoiceXML allows the developer to create scripts with dialogues that can incorporate spoken input and TTS or prerecorded audio for output [DDJY]. JavaScript for example, can be used at certain points within the page to direct flow and perform calculations. SALT could be regarded as an alternative to VoiceXML. Nevertheless, it is the author's opinion that, at this time, VoiceXML and SALT are separate and are being used for different things. It is also the author's opinion that enterprise use of SALT will likely be for the development of multimodal applications for areas such as Web-based self-service and call centres, etc. SALT-enabling a Web site would require additional Web development tools such as: tools that support the specification and SALT-capable browsers. Table 2.7 outlines some of the key differences between VoiceXML and SALT, which might help when deciding which technology to implement in an application:

VoiceXML	SALT
Work began in March 1999. Final Specification was due in the Autumn of 2002.	W3C work initiated in July 2002, with a final specification due in another 12 to 18 months.
Numerous shipping products.	Only a handful of early and beta products available.
Designed for telephony applications.	Designed for speech-enabling Web pages and multimodal applications, but can be used to create telephony applications.
Was regarded as a brand new language for telephony applications.	An extension to HTML, XHTML ²⁴ and XML, which should lower development costs.

Table 2.7 VoiceXML versus SALT [CSLT]

2.17 Data Transfers

With regard to HTML, the focus has been on presentation with data bound directly into the presentation. Conversely, in distributed applications the focus is placed on separating the display and the data delivery [SOPS]. XML is becoming the preferred way to supply data to client applications. XML has quickly become a forerunner as a

²⁴ Extensible Hypertext Markup Language

messaging format that serves as an intermediary between the data and the consuming client application [SOPS]. XML is usually converted from some native data format such as a database table or an object and is subsequently utilised as the transfer mechanism. Consequently, the client has the option of overriding the data directly through the XMLDOM or by converting it back into a native format such as a table or object that maps the XML.

XML may be used as a persistence format to transfer a state from the server to client or vice versa by changing it back into a native format. An XML parser is then required along with a mechanism for transferring XML across the network-using Hyper Text Transfer Protocol (HTTP) [SOPS]. XML is used as a storage or interchange format for data that is structured, appears in a regular order and is most likely to be machine processed instead of read by a human. In a data-centric²⁵ model, data is stored or transferred as XML. Depending on the data content and how it is utilised, data can be stored and/or transferred in various other formats which may/may not be better suited for the task [XDBO]. An example of a data-centric usage of XML is the Simple API for XML (SOAP). SOAP is an XML based protocol used for exchanging information in a decentralised, distributed environment [XDBO]. A SOAP message consists of three parts [XDBO]:

- An envelope that defines a framework for describing what is in a message and how to process it.
- A set of encoding rules for expressing instances of application-defined data types.
- A convention for representing remote procedure calls and responses.

²⁵ A model where data is stored in a relational database or similar repository.

SOAP does not alter the model, rather it standardises it in order to make remote calls on object methods. It can be stated that SOAP defines how the XML data can be transmitted from one point to another [DDJK]. In traditional XML applications, it is necessary for both client and server sides, to have prior knowledge of what the message format is, which results in coupling between the client and the server. SOAP can summarise the explicit XML conversions that occur in custom XML implementations by providing a standard mechanism for representing the procedure call interface and a mechanism for querying what functionality is available and what the signature of each call is [SOPS].

2.18 XML and Databases

As a natural progression, data becomes old and the software can become difficult to use as technologies change. As a result legacy data is unavoidable. There are companies employing the Common Object Request Broker Architecture (Corba), Enterprise Java Beans (EJB), or the Distributed Component Object Module (DCOM) in order to transfer data across from legacy systems [ITPC]. However, problems arise in that these technologies are code-centric. In other words the software must be part of a specific program infrastructure before integration can take place [ITPC]. It is for this reason that many means of integrating legacy code/data are limited. Nevertheless, XML eliminates the necessity to conform to a specific infrastructure as it describes data through tags. Although, two other means of transferring legacy code exist: Web services (refer to Section 2.20.1) and Wireless Technology, which facilitates the transferral of legacy data to mobile workers and clients [ITPC]. Nevertheless, XML is a much more useful technology as it gives data priority over code.

XML and JDBC (refer to Chapter 3, Section 3.6.4) work well together according to Mr. Kevin Williams in his book “Professional XML Databases”. By incorporating these two technologies it is possible to create ‘universal data access’ [PDBW]. These applications can run on varying J2EE-compliant application servers and access a variety of different data sources. The use of XML allows a developer to make the data and metadata available to almost any application regardless of the platform that it runs on.

XML data can be stored in a Database Management System (DBMS) that supports manipulation of and access to XML documents. Sample code and diagrams are given in Chapter 3 and Chapter 4. Figure 2.17 illustrates a three-tier architecture (refer Chapter 4, Section 4.3.1.1) for a Web database system:

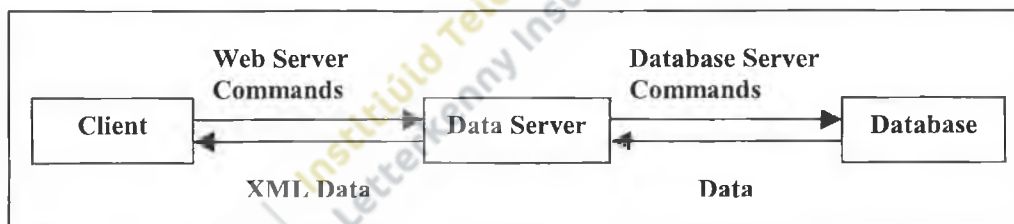


Fig. 2.17 Three-tier architecture for a Web database system

Loading XML data into a relational database that already exists can result in two main challenges (i.e. a semantic and a technical challenge) [DXDB]. Trying to map a ‘person’ element type in the XML to a personnel table in a database may prove problematic. There may be tags in the XML document that do not map to any concept in the relational database. The second challenge is to take a flattened file for storage that has been taken from the hierarchical XML data and put it into a relational database.

When XML-based information is successfully stored in an Oracle database, it can be searched, processed and presented in a useful way. Many Internet companies already have Oracle as a basis for managing their information for the Web. Consequently the Oracle8i/XML combination is a good means of furthering a company's business potential. It provides a sound basis for them to further profit from opportunities where E-Commerce is in use. Although the Oracle8i/XML combination has many potential uses, research conducted in this field revealed Oracle8i to be cumbersome and difficult to incorporate (especially where user permissions are an issue).

Microsoft Access 2000 however was less complicated to incorporate. Chapter 4, Section 4.3.2 describes how to connect to an MS Access database and retrieve the data as XML. The relevant ease at which a connection could be made with an Access database suggests that this DBMS would be a more appropriate option for a small business wishing to avail of the advantages that XML has to offer legacy data. Overall, there are less administrative tasks to be carried out.

2.19 Extensible Style Sheet Language Transformation (XSLT) and XML Path Language (XPath)

XML data can be stored, manipulated and managed. The parser used to manipulate the data is a program that can read XML syntax. By using a parser the finished application will never have to look at XML for the information. The XML parser that comes with Internet Explorer 5.0 also includes an XSLT engine [BXML].

XSLT and the XML Path Language (XPath), provide a powerful implementation of a tree-oriented transformation language for transmitting pages of XML using one vocabulary into the legacy HTML vocabulary, simple text, or XML instances using any other vocabulary. XSLT, which itself uses XPath, is used to specify how an

implementation of an XSLT processor is implemented to create the preferred output from a given marked-up input. Therefore, XSLT enables interoperability. By using XSLT it is possible to turn an XML document into HTML, XHTML or another form of XML [DDJN]. XSLT gives a programmer the ability to update style on a Web site instantaneously for thousands of managed documents [ACMS]. The following are a few examples of the roles that XSLT can play in the architecture of a system [PDBW]:

- XSLT can be used to format information for display purposes.
- In addition, XSLT can be used when managing data interchange between various computer systems. For example, part of an E-Commerce exchange with customers or suppliers, or simply application integration within the enterprise.
- XSLT can perform some of the roles that were usually carried out by report writers and 4th Generation Languages (4GLs). For Example, tasks such as information selection, aggregation and exception handling.

XPath is string syntax for constructing addresses to the information found in an XML document. This language is used to specify the locations of document structures or data found in an XML document when processing that information using XSLT. The XPath standard has been growing in its capabilities and it is fast becoming one of the main languages used to find data within an XML document [DBMSB]. Data models play a fundamental role in displaying the data within a document logically as a tree structure. This data model is used by XPath to retrieve the requested data. The tree consists of nodes for every kind of construct that can appear in a document [DBMSB]. The seven node types are [DBMSB]:

1. Element - There is an element node for every element in the document.
2. Attribute - Each element node has an associated set of attribute nodes.
3. Comment - There is a comment node for every comment, except for any comment that occurs within the document type declaration.
4. Namespace - Each element has an associated set of namespace nodes, which is implicitly declared by the XML Namespaces Recommendation.
5. Processing instruction - There is a processing instruction node for every processing instruction, except for any processing instruction that occurs within the document type declaration.
6. Root - The root node is the root of the tree.
7. Text - Character data is grouped into text nodes.

XPath allows any location to be addressed efficiently. XPath is a language that can be used for addressing parts of an XML document that uses a syntax, which is similar to that which is incorporated by hierarchical paths in order to address parts of a file system for example. In addition, XPath supports the use of functions for interacting with the selected data from the document [XDBO]. Functions are also available that are used for accessing information in relation to document nodes and for handling strings and numbers, etc. Furthermore, XPath is extensible in relation to functions, which facilitates the addition of functions that can manipulate the retrieved data by an XPath query to the library of functions available by default [XDBO]. Generally XPath operates on a single XML document, which it views as a tree of nodes.

Two vocabularies specified in separate W3C Recommendations provide for the two distinct styling processes of transforming and rendering XML instances. Information can be transformed using one vocabulary into an alternate form by using the

Extensible Stylesheet Language Transformations (XSLT). The Extensible Stylesheet Language (XSL) is a rendering vocabulary describing the semantics of formatting information for different media. By using XSLT and various other transformation tools Web designers can automatically produce Web pages that are appealing and can also create advanced multimedia shows [XSLTI].

XML namespaces are then implemented to distinguish information when mixing multiple vocabularies in a single instance. Without namespaces the processes would find the information ambiguous when the designers of the vocabularies used have chosen identical names.

2.20 Business-to-Business & Business-to-Consumer Applications

As a result of XML being relatively uncomplicated, it is changing the way in which software is created and utilised. XML Web services are now becoming part of a model that is used to develop applications. This model assumes integration to be a central part of the development process [SDPN]. With these Web services in place businesses can now use the Internet as a means of developing their business further.

2.20.1 XML Web Services

The XML Web services architecture is based on the principles of connection, communication, description and discovery [SDPN]. XML is a common data format that facilitates a connection by providing a standard configuration for data to be shared that does not necessitate that business partners or consumers use a certain programming language, application or operating system to interact with the system [SDPN].

XML is concerned with metadata information and so it is essential to describe the functions an XML Web service performs. With these principles in place a programmer can deliver XML Web services across the Internet or an intranet regardless of the programming language, computing device or object model used [SDPN].

With the advent of XML, several User-Groups have started to define their own markup languages such as MathML, etc. In designing such languages, they are defining ontologies²⁶ and shared vocabulary of their own domain. Ontology is now accepted as a fundamental building block of Business-to-Business (B2B), Business - to-Consumer (B2C) Communication and E-Commerce [BCMI]. Extensive research was carried out within the field of B2B applications. It is becoming one of the fastest growing aspects of the World-Wide-Web today.

B2B generally involves a relationship between the supplier and a purchasing company (in other words between two businesses). In a recent survey²⁷, Forrester Research estimated that B2B infrastructure will power Europe's online trade to €2.2 trillion in 2006 [FBTB]. The World-Wide-Web (WWW) offers an attractive method of enabling a wider, more streamlined approach to getting products and services to customers in a much faster way. XML provides a means of creating B2B applications efficiently. An example of a B2B application would be Microsoft's 'Microsoft Market' [BXML]. It was created to reduce costs for Microsoft. It was developed to allow employees at Microsoft visit this 'Market' and buy products cheaper online. This B2B application

²⁶ A means of exchanging structured information

²⁷ 2002

has meant that costs incurred by Microsoft have reduced from \$60 per transaction to just \$5 per transaction.

B2B applications only really exist between companies that have pre-arranged or negotiated a deal. For example, when an employee orders a product, an E-mail (Electronic-mail) is sent to the employer to confirm. When it has been confirmed only then will the transaction go through. A B2C application is less secure in that a valid credit-card number will suffice in order for a transaction to go through. A relationship does not exist between the buyer and seller. No negotiation takes place.

XML attempts to make information self-describing, thus helping to solve one of the Internet's biggest problems. For example, the Internet is an extremely fast network that often slows to a crawl and although nearly every kind of information is available online, it can be very difficult to find the one piece of information required. This problem is partly due to the restrictions of HTML, the Internet's main language. HTML is superficial in that it is mainly concerned with the appearance of text rather than with what the text signifies. This means that when an item is ordered, a remote server has to process the information, because the HTML-powered web page cannot handle this kind of transaction by itself.

There are certain tasks that could be made a lot easier such as arranging a meeting with colleagues from partner companies but they are proving difficult to organise. In the future XML will be used to alleviate this problem [SDPS]. Because XML separates the original data from how that data is displayed, then the data can easily be structured, programmed, condensed and exchanged between various different Web sites and devices [SDPS]. The Web was once used to provide a means for users to talk to applications, now XML is being used to enable applications to talk to each other

[SDPS]. Table 2.8 shows a table of Microsoft's products and services that currently support XML.

	Present	Future
Client Operating System	Windows 2000 Professional, Windows ME, Windows CE	Windows XP Professional, Windows XP Home Edition, Windows XP Embedded, Windows CE 'Talisker'
Smart Devices	Pocket PC, Mobile Explorer	Xbox, Tablet PC, Smart phone, Ultimate TV
User Experiences	MSN Explorer, Office XP, Visio 2002	Next version of Office, Next version of MS Project, Next version of Visio
Building Block Services	Passport	HailStorm Services
Developer Tools	Visual Studio 6.0, SOAP Toolkit 2.0, Visual Studio.NET Beta 2, .NET Framework Beta 2	Visual Studio.NET, .NET Framework, .NET Compact Framework
Servers	Windows 2000 Server, Windows 2000 Advanced Server, Windows 2000 Datacentre Server, SQL Server 2000, Exchange 2000 Server, BizTalk Server 2000, SharePoint Portal Server 2001	Next version of Windows, Next version of SQL Server

Table 2.8 Products and Services [SDPS]

A B2B application makes information about products accessible globally and also updates the information in real time [BCMI]. This adds to the reliability and speed of purchasing services on the Web. When XML is utilised in B2B applications the end result tends to be a much faster system. With standards such as XSLT and XPath being utilised on a more daily basis, the means of displaying and manipulating such data is becoming more readily available.

2.20.2 Legal Issues

A study has been conducted that concluded that most websites are not accessible to the disabled and new laws to enforce standards should be introduced. A 2-year study

of Irish websites, conducted by a team at Dublin City University discovered that most of the 159 websites tested failed to meet the minimum accessibility standard for disabilities as defined by the W3C [DSBM].

The study also discovered for example that 98% of the sites used rigid “pixel perfect” displays, which created barriers to magnification, which made access more difficult for visually impaired users [DSBM]. These facts affect the accessibility of the Internet and it is the author’s opinion that XML technologies can help resolve some of the issues that exist on the Internet (refer to Chapter 3, Sections 3.3.4 and 3.5). ‘The Harris Poll’ survey carried out in 2000 also revealed that adults with disabilities spend, on average, twice as much time online as adults without disabilities [HPWD].

The author of the aforementioned study, Dr. Barry McMullin, stated that it should encourage government, public agencies, private companies, organisations and individuals to take measures to ensure that standards are raised [DSBM]. It is the author’s opinion that XML could help increase the capabilities of websites with the advancements of SMIL technologies and VoiceXML for example (discussed in Sections 2.13 and 2.16 respectively).

2.21 XML and the Future

The applications that will aid the acceptance of XML are those that cannot be achieved by HTML due to its restrictions. These applications can be separated into four extensive categories [HTMLB]:

1. Applications that require the Web client to mediate between two or more varying databases.

2. Applications that attempt to distribute a significant proportion of the processing load from the Web server to the Web client.
3. Applications that require the Web client to present different views of the same data to different users.
4. Applications in which intelligent Web agents attempt to tailor information discovery to the needs of individual users.

The alternative to XML for certain applications is code embedded as script elements in HTML documents and delivered in conjunction with browser plug-ins or Java applets [HTMLB]. With XML the data belongs to creator and the content providers are provided with a data format that does not limit them to using certain script languages, authoring tools, and delivery engines but provides a standardised, vendor-independent, environment where different authoring and delivery tools may compete without constraint [HTMLB].

XML and Java technologies are united by a common principle, that is, platform independence. Collectively, the two technologies are enabling a new generation of web applications in areas ranging from Electronic Data Interchange (EDI) to E-Commerce and workflow management to enterprise resource planning. Wherever there is a need for information exchange on network systems, XML and Java technologies are the best possible choice. The fact that Java supports Unicode gives it an advantage over other programming languages. While older languages such as C, C++ and Perl were built around one-byte character encoding, Java was created after it was deemed necessary that the two-byte Unicode character-encoding scheme was extremely useful when creating programs [BXAL].

As a result of the Java platform's portable code capability, the technology has been very important in the development of a collaborative environment [XJAV]. For example, the XML-Dev²⁸ mailing list incorporated Java technology as the basis for creating a project called SAX. SAX is a Java technology that permits applications to combine with any XML parser to obtain notification of parsing events. Most major Java technology-based parsers now support this interface [XJAV].

The following are other key features that the Java platform shares with the XML standard [XJAV]:

- The Java platform fundamentally supports the Unicode standard, which makes processing an international XML document much easier. For platforms that do not have native Unicode support, the application must incorporate its own method of handling Unicode characters, which inevitably adds complexity to the overall solution.
- The Java technology facilitates a highly productive environment for processing and querying XML documents. The Java platform can become an ever-present runtime environment for processing XML documents.
- Java's platform has natural support for object-oriented programming, which means that developers can create applications by developing hierarchies of Java objects. In the same way, the XML specification presents a hierarchical representation of data. For that reason, Java and XML share a common underlying feature, which indicates that they are compatible for representing each other's structures.

²⁸ XML-DEV is used for the productive discussion of XML topics.

In order to take full advantage of XML, Sun's Java Server Pages (JSPs) and Enterprise Java Beans (EJBs) technologies can be used for building applications ranging from stock tracking systems to tools that track employee benefits and sales commissions [JAVB]. The flexibility of XML's metadata and data portability give Java technology a considerable advantage in making data more portable over a network, while the Java language improves productivity in comparison with C and C++. XML and Java technologies are both Unicode²⁹ aware, easing the way for international commerce [JAVB]. The two combined also help preserve access to legacy applications. By means of connecting with existing databases and distributed applications, as well as supporting native APIs and classes, the applications written for a Java platform interoperate with productivity applications and legacy systems [JAVB].

2.22 Conclusions

XML is similar to HTML's design but offers users a more extensible language. It lets information publishers create their own tags for applications. Alternatively, they can work with organisations to define shared tag sets that promote interoperability and help separate content from presentation [XDBI]. While XML addresses content, the CSS, the XSL, and XHTML handle presentation separately. XML also supports data validation. XML's advantages over HTML include support for multiple views of the same content for different user groups and media; selective, field-sensitive queries over the Internet and intranets, a visible semantic structure for Web information, and a standard data and document interchange infrastructure [XDBI]. Using XML and related tools often reduce problems associated with heterogeneous data structures. However, the need for database management systems will still exist.

²⁹ Unicode is the World's standard for encoding text.

There are many advantages that XML can offer to improve networking capabilities on the Internet (e.g. XSLT (caching), SVG (file size reduction), etc.) but there are also disadvantages, primarily lack of support. Advancing the Web can only be welcomed but steps need to be taken to ensure that these 'new and improved' technologies are actually catered for. Having capable applications created in SVG and SMIL for example, may advance the Web with respect to providing information to a wider range of people more quickly, but the application is useless if the browser does not support it. It can be noted that almost all of the XML applications covered in this Chapter require either plugins or software in order to run successfully.

A VoiceXML file that was created in order to determine the validity of a user's credit card revealed it to be 4,351 bytes in size. If this application were to be utilised in a working environment (i.e. not using the IBM WebSphere Voice Toolkit), it would generally require two voice lines to be installed (as discussed in Section 2.16). This would inevitably increase the cost of presenting a voice-driven system.

XML is becoming widely used, especially in the E-Commerce industry. Consequently, its interaction with two databases (MS Access 2000 and Oracle8i) was analysed. As XML is a good language for data exchange, it is often used in communication between systems. Database system architecture describes how applications and users access and manage data in a DBMS. Data stored relationally in a database could be retrieved and displayed within the application through an Application Server (Tomcat) and given additional semantic meaning. Thus, legacy data could be manipulated by a new technology and successfully viewed.

As a result of the literature survey conducted many questions were identified regarding XML's capabilities (or lack thereof). It is therefore necessary to design and develop applications that would further investigate issues such as:

- The undocumented repercussions of XML when interacting with legacy databases.
- XML's ability (or inability) to act as a database itself.
- XML's formatting capabilities.
- XML's ability (or lack thereof) to advance Web sites/CBTs, etc.
- The potential portability problems of SMIL.
- The need to install additional hardware/software/plugin in order to successfully run applications of XML.
- The lack of support available for XML.

It was for these reasons that it was deemed necessary to create prototype applications that would test these capabilities and enable evaluations to be conducted. Chapter 3 outlines the design for the prototype applications.

Chapter 3

Designing the Prototype Software

lyit

Institiúid Teicneolaíochta Leitrí Ceanaínn
Letterkenny Institute of Technology

3.1 Introduction

Software design is a significant phase in the development of any system. If the design is carried out in an appropriate way, this may alleviate many programming problems that might otherwise arise. The use of quality software design allows for the correction of any poor functionality within the system on a regular basis. It also helps the designer make the best possible use of the code being utilised. E.S. Taylor¹ once defined software design as: good

'Design is the process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realisation.' [RPTT]

Whilst keeping this definition in mind the applications created were developed in accordance with a number of design methodologies, which are discussed in more detail in this chapter.

3.2 System Requirements

One objective of this research is to provide an investigation into the interoperation of XML and the capabilities that this language has to offer. For the purposes of this thesis only a few aspects of XML were explored in the prototype system. These aspects include:

VoiceXML,

SVG,

SMIL and to a lesser extent,

MathML and VML.

¹ Massachusetts Institute of Technology, Cambridge, MA

The system created also considered how XML interacts with legacy databases. To create a more realistic system another language had to be included (in order to connect to the database, etc.), in this case an application incorporating Java was deemed necessary. For this reason the following points required careful consideration:

The ability of Java and XML to interoperate.

The viability of connecting to a database using JDBC.

The possibility of storing the data in a relational database.

The need to run the application locally on the machine incorporating an application server.

Keeping the above in mind, two prototypes were developed which incorporated a number of features. The first application developed, implemented the applications of XML that could run successfully on Internet Explorer 5.5 or higher. It was necessary to prove that these applications could interact with each other thus demonstrating their capabilities (i.e. animation, embedded SMIL tutorials, mathematical notations, graphics, etc). Section 5.7 in Chapter 5 discusses in detail the problems that arise when trying to amalgamate these various languages, as support is still quite limited. The primary application (Sigma-X) was developed using SVG as a basis for designing the interface (refer to Chapter 2, Section 2.12). This application also provided the means to prove that XML was a useful asset in the manipulation of legacy data. XML was utilised to provide additional semantic information to data stored in relational databases such as MS Access. Java acted as a means of uploading core data into a database. Section 2.21 in Chapter 2 and Section 4.4 in Chapter 4 discuss why the use of Java with XML proved to be a more viable option than incorporating other object oriented languages.

The second application was developed to prove that XML was a useful asset in the manipulation of voice activated Web sites. From the research conducted into the area, it was concluded that an ever-increasing importance is being placed on the ability of Web designers to incorporate 'voice' into their Web sites (refer to Sections 3.5.1.2 and 3.5.1.3). For demonstration purposes a small application was developed in VoiceXML, which was run using the IBM WebSphere Voice Toolkit. The application enabled the system user to interact with the menu using voice driven commands. Incorporating this application into a Web site proved difficult at the time of this writing, as support for SALT tags (refer to Chapter 2 Sections 2.16 and 2.16.3) was relatively unknown.

3.3 Design Specification

'There are two ways of constructing a software design: one way is to make it so simple that there are obviously no deficiencies and the other way is to make it so complicated that there are no obvious deficiencies'. C.A.R. Hoare [CARH]

The Design Specification was created with considerable attention to a number of points, including:

3.3.1 Scope of the Application

Particular attention needed to be given to the nature of the system and the purpose of its development. The applications demonstrate a number of the capabilities of XML and show how XML can be used to interact with Legacy Databases, by creating a series of mini-applications within the system that attest to the capabilities of XML.

One of the most common E-Businesses on the Internet today is that of E-Learning applications. The nature of E-Learning requires many different data types (e.g. video, text, images, sound, etc.) to be utilised in various degrees across E-Learning programs. Thus, it was decided to create an E-Course through XML technologies.

3.3.2 Data Design

The type of data to be accessed by the applications required consideration when designing the system. However, greater consideration was given as to where the data would be stored and how it would be accessed. The data would be stored relationally in a table in MS Access and retrieved accordingly using the JDBC-ODBC Bridge. In the Oracle system, the tables would be constructed by executing a series of SQL statements that would enable the system to traverse the XML document, retrieve the required data and place it in the relevant tables when uploading into the database using a Thin Driver.

3.3.3 Designing the Interface

Human-Computer Interface (HCI) guidelines were followed when designing the interface to the systems in order to create an application that was aesthetically pleasing and viable. When considering the interface design, it is important to note the type of persons that would use the system. These applications are intended for use within this Third Level Institution only. Participants within the Institution would also vary from computer experts to computer novices. The age of the application user would also affect the overall design, which may vary from a young adult to someone in their sixties. Due to the nature of the research, the functionality of data processing

was given priority over aesthetics when necessary in order to successfully demonstrate the capabilities of XML when interacting with databases.

3.3.4 User Interface Design Considerations

The following are factors that relate to designing a good user interface [SMND]:

Support both novices and experts: The system must also be accessible to novices and experts. The tutorials in the system must be written in a way that even someone who has never heard of SVG for example, could learn how to program in it or learn a little of the history and understand it. However, these tutorials must also cater to the needs of someone who has heard of XML and wishes to expand their knowledge. The same would apply when connecting to the database. Not everybody has used a database.

Navigation: If it is difficult to move from one screen to another then the system users will quickly become frustrated. When the flow between screens matches the flow of the work that the user is trying to accomplish, then the application will make more sense to the user. It is also important to consider that different people work in different ways, which implies that the system will need to be flexible enough to support their various approaches.

Wording: The text displayed on the screen is a primary source of information to the user. If the text is worded poorly then the interface might be considered poor. For example, when relaying information to the user on SVG or SMIL, it might be better to use full words or sentences rather than abbreviations. How messages to the user are displayed are also important factors and the consistency of those messages displayed.

Use Colour Appropriately: Colour should be carefully considered in applications (i.e. limit to a small number of different colours) and where colour is used a secondary indicator must also be used. Consideration must be given to certain users that may have sight disabilities (such as colour-blindness). When taking into account these disabilities, it might be useful to place a symbol beside text, buttons, or a picture that would normally be highlighted with colour. It is also an important consideration that colour sometimes does not port well between platforms.

Contrast Rule: When colour is being used in the application it is important to ensure that the screens are readable. A good rule-of-thumb to incorporate is: use dark text on light backgrounds and light text on dark backgrounds.

Use Fonts Appropriately: When designing this application it was important to consider the fonts that were to be implemented. It is necessary to use fonts that are easy to read, such as serif fonts like Times Roman. These fonts should also be used consistently and sparingly. Please refer to the study by Barry McMullin in his article "Users with Disability Need Not Apply? Web Accessibility in Ireland" [DSBM].

Use Buttons Appropriately: The buttons should be designed in accordance with the rest of the application and be placed consistently on the screen throughout. Also, it is important to note that a default button should not be a button that will carry out a potentially destructive operation such as delete or save.

3.4 Hardware/Software Mapping

The hardware and software required careful consideration and would result in additional requirements having to be incorporated into the applications. This is further discussed in Chapter 4. Deciding which database systems to use also required consideration as it was necessary to demonstrate how XML could be advantageous to both large companies and small to medium businesses. Hence, Oracle and MS Access were incorporated accordingly.

3.5 Global Resource and Access Control

It is becoming increasingly important to create Web sites that are accessible to people with disabilities. It is the primary objective when developing a web site to help remove or reduce barriers that might affect the daily activities of someone with a disability. The four main categories of disabilities (but not exclusively) are visual, auditory, mobility, and cognitive and learning disabilities [WMEI]. Web developers can alleviate some or all of these barriers Web site design is more carefully considered. An article by Christine Maxwell put forward the theory of access to the Internet: for all people including those with disabilities is tied to all other issues concerned with establishing a global and affordable Internet that can be used by everybody [MCRI]. This thesis investigates the capabilities of XML, therefore it is necessary to discuss how XML and applications of XML (namely SVG, VoiceXML and SMIL) can help make information available to as wide an audience as possible.

3.5.1 Relating Disability Issues to the System

It was necessary to develop the applications to conform to standards and be accessible to as wide an audience as possible. Exploring the capabilities of XML is the primary goal of the applications to be developed however accommodating people with disabilities would make the software available to more people on-line which is one of the main goals of E-Commerce.

3.5.1.1 Visual

People with visual disabilities are individuals who are blind, have low vision, or have colour blindness. When providing for people with colour blindness it is best to use contrasting colours according to an article by Chuck Newman [NCCB]. However when information is presented using only colour then someone with a visual disability may miss the information that is required. Therefore, utilising the correct colour scheme can be very important. For example it can be considered a good technique to combine 'cool electric' colours with 'warm' colours (i.e. green background requires magenta/purple text, beige/yellow background requires blue/violet text) [NCCB]. However, an alternative to presenting the information may also be an answer. SMIL may be incorporated to help alleviate the problem as the information can be displayed using colour and text as well as having a spoken dialogue relay the information to the user. This way if a user has visual disability then they have a better chance of retaining the information.

By magnifying the information, a user who has only partial site may be able to read text more clearly or view graphics more distinctly. This is where the pan and zoom capabilities of SVG may advantageous. The text/graphic can be increased in size

without distorting the image thus providing a means of making the information available to someone with a visual disability. Such XML technologies can extend the possibilities of the Internet to a much wider audience.

3.5.1.2 Auditory

Providing a Web site that caters for people with hearing difficulties requires a similar approach to that of providing accessibility to those with visual impairments. Where information is relayed using audio means, then a textual alternative is required. SMIL could also be advantageous in this sense as information is relayed on a given topic through spoken dialogue but is accompanied by a collection of slides that present the same information textually.

3.5.1.3 Mobility

Users with mobility disabilities generally have physical impairments that affect how they can move, inevitably restricting how they can interact with a system through a mouse or keyboard. It is the author's opinion that VoiceXML could help such users. Solutions for persons with mobility disabilities include alternate input capabilities, such as voice input or the ability to enter information at the user's own pace. Allowing the user to access information through voice commands could greatly improve their learning potential and enable them to access a much wider range of information.

3.5.1.4 Cognitive and Learning Disabilities

This is an aspect of the design that required considerable attention. People with cognitive or learning disabilities, such as dyslexia, need more general solutions, such as providing a consistent design and using simplified language. It is important for a

Web developer to incorporate a similar layout and design for each page, so a person with a cognitive disability can learn to navigate through a Web site a little easier. People with cognitive or learning disabilities can also benefit from redundant input [WMEI]. Redundant input implies that the developer provides both an audio file and a transcript of a video or spoken dialogue [WMEI]. Viewing the text and hearing it read aloud simultaneously, enables a person with cognitive and learning disabilities can take advantage of both auditory and visual skills to understand the material better. SMIL can offer such capabilities to users.

3.5.2 Unicode Standard & Global Distribution

A considerable advantage to XML is how it considers Unicode standards. Complying with Unicode standards implies that the documents contains additional characters for interoperability with older character encodings and characters that provide a clear interpretation of plain text. Unicode outlines how these characters can be utilised.

The Unicode Standard is a character coding system. Its purpose is to support interchange, processing, and the display of written texts worldwide thus facilitating many of the world's languages [MMUS]. This standard enables documents to be exchanged and accessed over the Internet regardless of the language and/or the country of origin. This is an aspect that was given considerable consideration when designing the application. Therefore, it was necessary to create an application that would access legacy data stored in a relational database, retrieve the required data and display the result in XML. The returned document could then be exchanged/accessed over the Internet if desired.

All of the aforementioned factors (i.e. disability accommodation, Unicode) can be accommodated with XML. These factors further emphasise the adaptability of XML for the global distribution of data. It was therefore, concluded that these factors would be taken into consideration when designing the applications based on XML. XML provides capabilities that enable users of varying abilities to access information globally through the added functionalities offered by SVG, SMIL and VoiceXML for example. Data can be formatted and accessed in various ways that imply that XML is helping to advance the distribution of data to a wider audience.

3.6 Subsystem Decomposition

It is necessary to break down the operations carried out within the system into smaller sections in order to reduce complexity.

3.6.1 Data Flow Diagrams

A Data Flow Diagram (DFD) is a graphical technique that illustrates information flow and the transforms that are applied as data move from input to output [SWEP]. A DFD may be used to depict a system or software at any level of abstraction. DFDs can also be split into levels that are representative of increasing information flow and functional detail [SWEP].

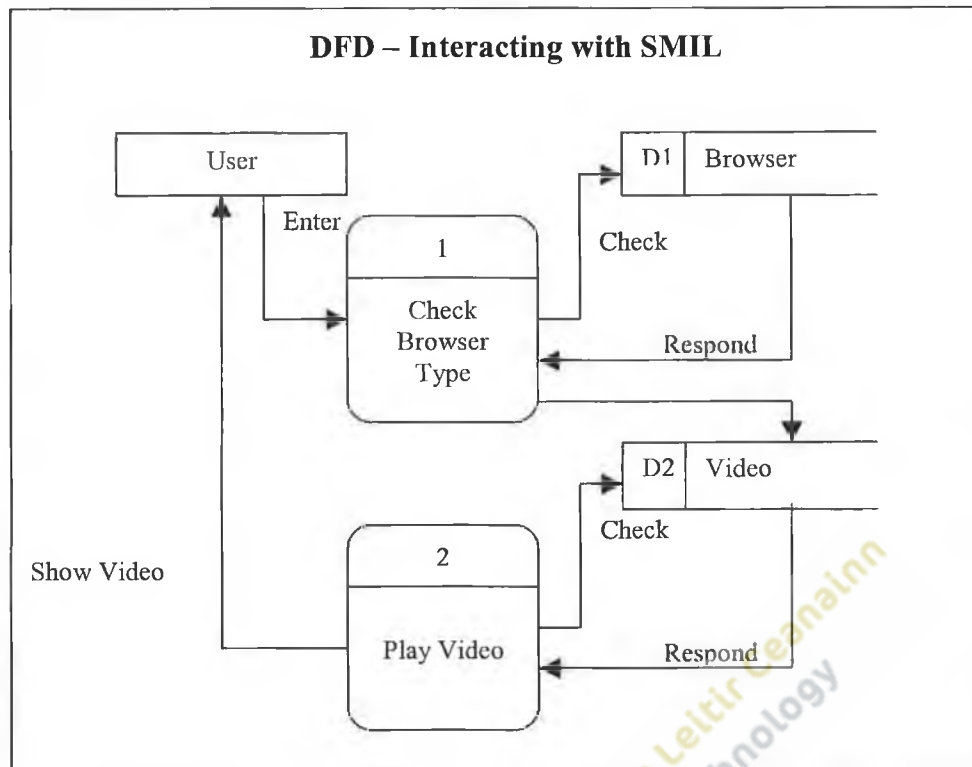


Fig. 3.1 DFD – Interacting with a SMIL Presentation

As can be seen in Figure 3.1, the user can interact with a SMIL presentation by requesting a tutorial on a given topic. If the tutorial can be displayed on the user's browser then the appropriate response will be given. The user can also decide if they would prefer to watch a video on the topic or view a presentation in the form of graphics and spoken dialogue.

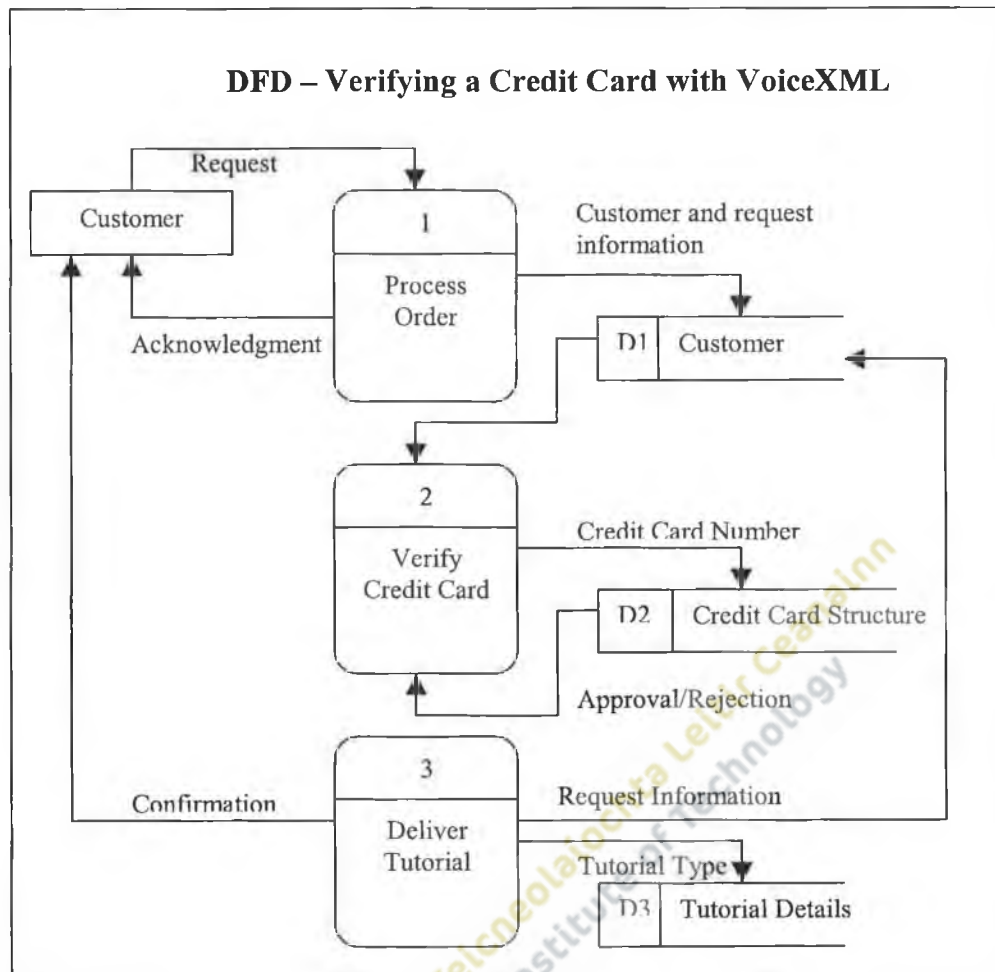


Fig. 3.2 DFD – Verifying a Credit Card with VoiceXML

In Figure 3.2 above a person's Credit Card details are verified using VoiceXML. Firstly a Customer places a request by relaying their credit card number to the system and the Customer's details are stored in data store 'D1'. An acknowledgement is sent to the Customer reassuring them that the order is being processed. The credit card details are then verified and depending upon the credit card being valid or invalid, the details are stored in 'D2'. If the order is accepted then the order information is stored in 'D1' and the tutorial type is stored in 'D3'. A message of confirmation is then sent to the Customer along with the requested tutorial.

3.6.2 Unified Modelling Language (UML)

UML was chosen because it is a popular method for designing software and has demonstrated that it is a valuable means of modelling data. Other additional reasons why UML was chosen are as follows [CMPS]:

It is a well-used standard that is well known to many software developers, widely taught in undergraduate courses, and supported by numerous books and training courses.

Many tools from different vendors support UML (such as Rational Rose). There is also an excellent conceptual match between the object paradigm and real-time systems.

An additional benefit of UML is that it is extensible, using stereotypes in a UML Profile [ACMR]. This implies that a method for designing XML Schemas can be developed, which is compatible with existing UML tools [ACMR].

The development of UML, arose as a result of the combined efforts of Grady Booch, Ivar Jacobson and James Rumbaugh, is representative of one of the most important advancements in object technology [CMPY]. UML defines the semantics of object model elements and their notations incorporated by popular analysis and design methodologies [CMPY]. Exchanging models is an integral part of today's environment and it is very important that the semantics within a model be described explicitly. A possible solution to the problem of exchanging models is, UML eXchange Format (UXF), which is based on XML and acts as a means of communication for transferring model information among development tools [CMPY].

Please note²

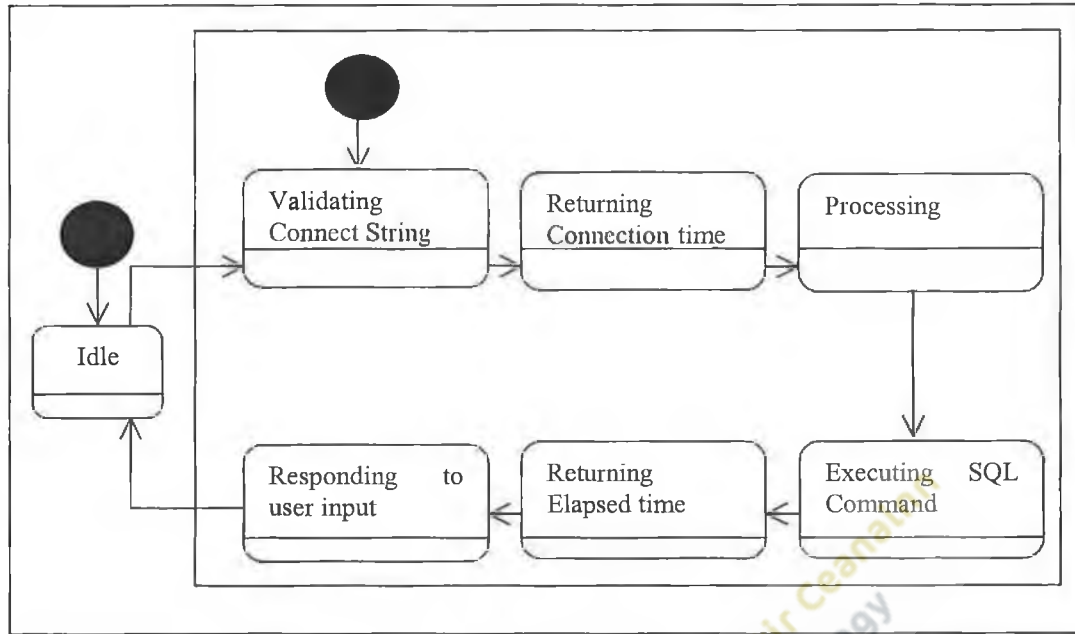


Fig. 3.3 UML State Diagram: Selecting a Menu Option in an Oracle Database

In Figure 3.3 a UML diagram depicts how a user might interact with an Oracle application by selecting an option. Initially there is no connection to the database but when a user selects a button a connection is made to the database and the time it took to connect is returned along with the connect string itself. Behind each button is an SQL statement (design consideration), which performs a manipulation on the data in the database. This SQL command is processed and executed. When the connection is complete, the time it took to execute the command is returned to the screen. The user can then make another selection.

The state diagram describes the behaviour of and the internal states within a class, focusing on how objects over time change their states, depending on events that occur, the actions and when transitions occur.

² An explanation of the notation contained in the diagrams within this chapter can be found in Appendix E – Code Keys.

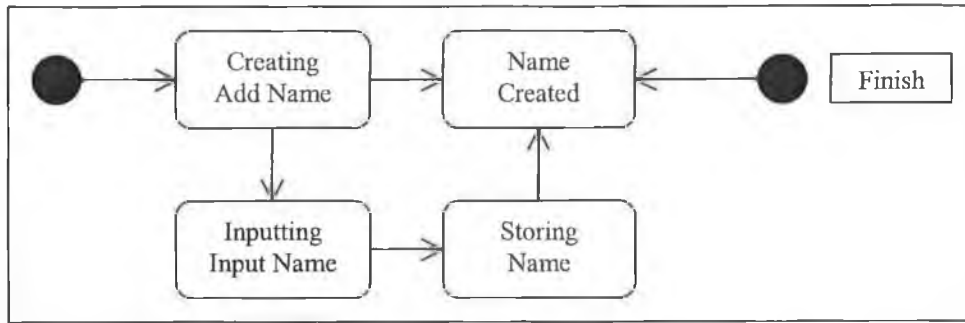


Fig. 3.4 UML Dynamic Model: Adding a Name Into an Oracle Database

UML Dynamic Model diagrams are used to model the interactions, activities and states of objects within a system. In Figure 3.4 a Name can be added to a database in Oracle. A user can click on a button and data in the database can be manipulated. In Figure 3.4, data is being added to the database.

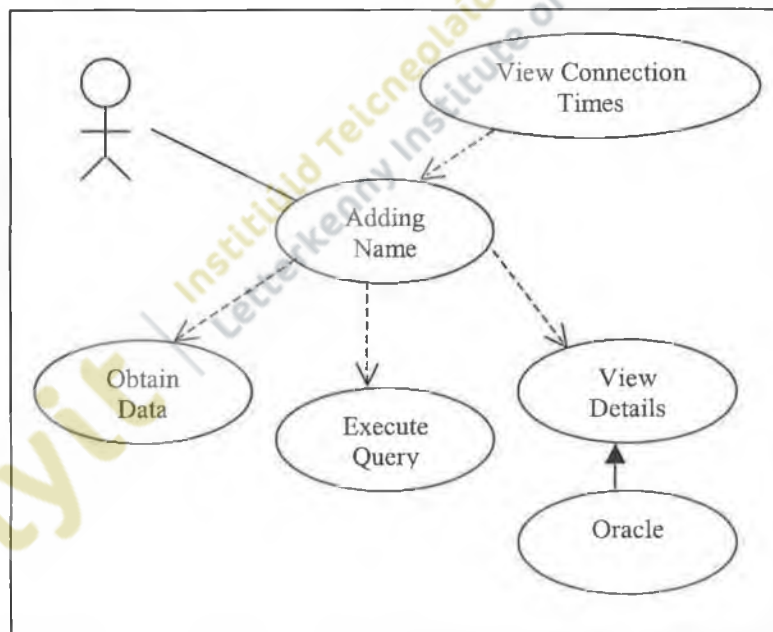


Fig. 3.5 UML Use-Case Diagram: Inputting Details Into Oracle

‘Adding Names’ required a Use-Case diagram to be used. This is necessary because the users of the system are identified and the tasks that they must undertake are also

identified. The customer, in this case (Actor³), represents how a user can interact with the system. Use-case diagrams are generally used during the requirements analysis stage. It depicts what the system should do and can then be used as a comparison to the final system to determine if the desired result was attained. Since use-case diagrams are not directly object-oriented, it was conceivable to incorporate this method of modelling certain aspects of the system that did not rely entirely on objects or classes. Figure 3.6 depicts how a user can interact with the database and what requirements are needed before the user can access the tutorials.

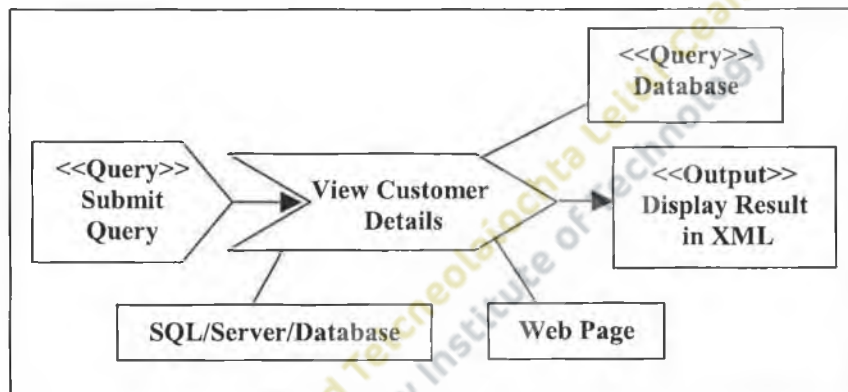


Fig. 3.6 UML Business Process Model: Viewing Details in MS Access

In Figure 3.6 the UML diagram depicts how a user can submit a query to a database in order to view customer details. In order to do this the system must have at its disposal SQL, a Server, a Database and a Web page on which to display the information. It can also be clearly seen that the goal of the system is to <<Query>> a Database. If the information has been processed correctly then the <<Output>> should be displayed in XML format. Figure 3.7 is a class diagram depicting the operations carried out by a Java program in order to upload an XML document into an Oracle database.

³ UML notation that is used to describe someone who interacts with a system.

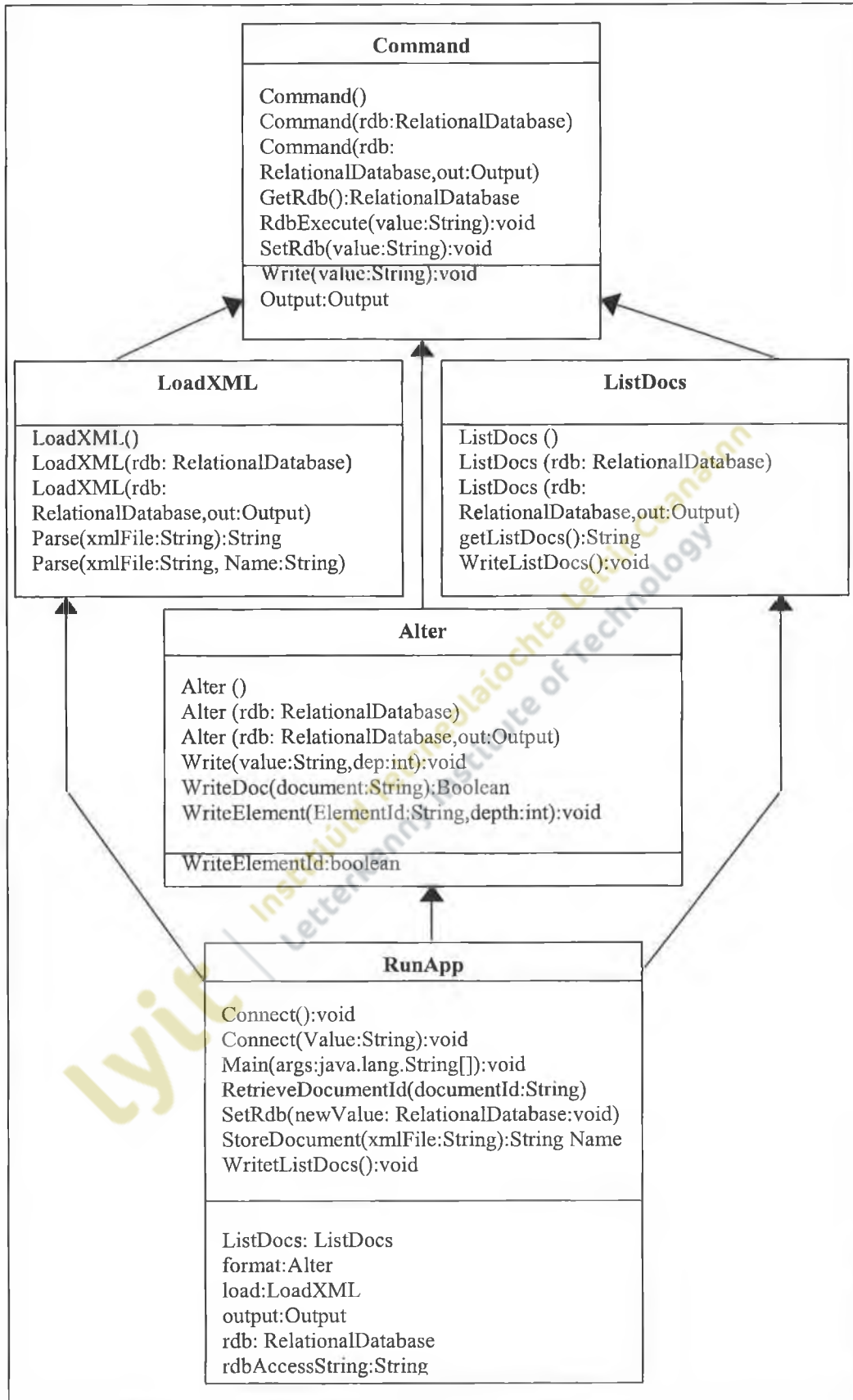


Fig. 3.7 Class Diagram - XML document being uploaded into Oracle

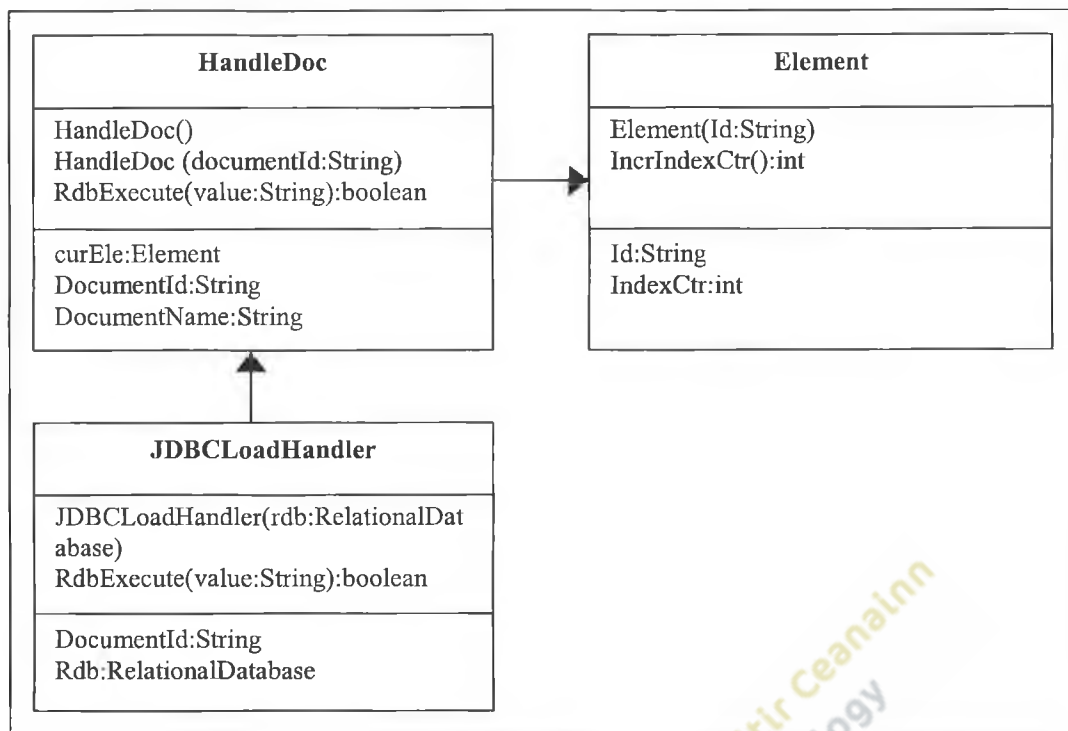


Fig. 3.8 Accessing the JDBC for Oracle

Figure 3.8 shows how the JDBC is accessed through the Java program

3.6.3 Storing XML in a Relational Database (Oracle8i)

XML data can be stored in a Database Management System (DBMS) that supports the manipulation and access of XML documents (i.e. searching, retrieval, inserting, updating, and deleting). Oracle8i and MS Access provided these capabilities. Java can be used as a means of manipulating this data but query languages such as XPath (refer to Chapter 2, Section 2.19) could also be incorporated. It is worth stating that XML DBMSs are still quite new and not as mature as relational or object-oriented DBMSs [DXDB]. For the purposes of this thesis, it was decided to store XML in a relational database.

A DBMS can be referred to in terms of its external, conceptual and internal layers [DXDB]. The external layer is used to provide support for the necessary functionality of the application. The conceptual layer exists to satisfy any data manipulation and querying requests and the internal layer is used for data storage. A typical DBMS structured can be depicted as follows in Figure 3.9:

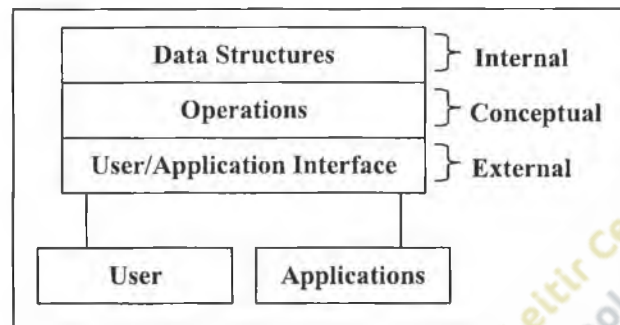


Fig. 3.9 DBMS Architecture

3.6.3.1 Available Storage Facilities

It is possible to store XML as a flat file in an object-oriented database or as a flat file in a relational database [DXDB]. The simplest way to store XML data is to store the whole document in a single file. As a result, it is easier to access it from any programming language and is also more readily accessible from all XML parsers. Flat file databases are a useful alternative to traditional DBMSs. Although XML uses this flat file structure, the additional semantic information that it provides help to resolve some of the flat file issues such as an increase in file size. A flat file can be as large as the operating system will allow or as small as a single file. When a file is small for example, then overheads that would normally be incurred by the DBMS can be reduced. However, the size of a flat file must remain relatively small for performance to be reasonable. The storage of information into an XML tree structure involves

more than twice the data size in management information. Thus, it would be efficient to store the data in a relevant database with the use of XML for format/display and distribution of information (refer also to Section 3.7).

3.6.3.2 Loading Data into Oracle8i

Loading XML data into a relational database that already exists can result in two main challenges, a semantic challenge and a technical challenge [DXDB]. Trying to map a 'person' element type for example, in XML, to a Personnel table in a database may prove problematic. It is possible for tags in the XML document to exist that do not necessarily map directly to any concept in the relational database. The second challenge is to take a flattened file for storage that has been taken from the hierarchical XML data and put it into a relational database. A solution to the problem would be to create a Java program that would read and parse an XML document and insert the data into the appropriate tables.

It was necessary to log into Unix as the Oracle Administrator, which gives ownership of a wide range of Oracle related software. It was then necessary to load the Transparent Network Substrate (TNS⁴) listener, check its status in order to retrieve the services it supports (STUDENTC) and start it (refer to Appendix G, to view the connect-string in the Java code). Finally the client was loaded and it was instructed to use the STUDENTC service, which it tested and found. It was also essential to assign STUDENTC as the host string when the Windows version of SQLPlus was loaded.

The developed application made use of several Java class files in order to upload an XML file into a relational database set up in Oracle8i by executing a batch file (which

⁴ TNS Listener is responsible for establishing and maintaining remote communications with Oracle database services.

specified the path to the XML file). Each tag in the XML document was checked and was allocated to a table in the database accordingly. Once loaded, the user could then manipulate the data as desired.

3.6.4 Retrieving XML from a Relational Database (MS Access)

As the use of XML is becoming increasingly popular, the need to be able to view relational data as XML is also growing [RDCO]. A series of Java programs were therefore written that would take an SQL command entered by the user, connect to the required database using the correct URL, username and password, etc. via the JDBC-ODBC Bridge. In order to retrieve data from the JDBC source and display the result as XML, the use of a Java class, which was called the 'JDBC2XML', was required. It was this class that controlled access to the JDBC data source within the application. It enabled a user to execute an SQL statement against a specified JDBC data source. The returned JDBC result set was serialised as XML. The XML was then returned to the client. This class was reused by a Java servlet called the 'XMLDataGateway'. The servlet provided a generic XML-over-HTTP interface to JDBC data sources for XML-enabled applications [PDBW]. Figure 4.5, "Servlet handling a request", in Chapter 4, Section 4.3.2.1 depicts this interaction.

3.6.5 Data Transportation

Transporting data was an important aspect of the system that needed consideration when designing the application. One of the main advantages to using XML is that it can be used to transport data as well as store it. However, a study carried out by Mark Graves (in his book "Designing XML Databases", refer to Bibliography for further details) took a simple XML document and determined what percentage of the

character spaces were taken up with tags as opposed to data itself. His study showed that only 15% of the document was taken up with actual data whereas the remaining 85% was taken up with tags [DXDB]. This study appears to refute the idea that XML is a Data Centric application. However, a Butler Group report, which was a Technology Evaluation and Comparison Report, stated that XML could be viewed as a form of database [DBMSB]. It is the author's opinion that XML is a form of database and through the separation of data from formatting, this allows XML documents to be designed in a data-centric manner. This would require further investigation in a larger system that would require more intensive operations with XML and databases.

3.6.6 Structure of the System

Figure 3.10 depicts the interactions between the various different languages in Application 1 (Sigma-X, refer to Section 3.9.1, Table 3.3). Table 3.1, outlines what each node number represents and a brief explanation as to what is being carried out at that particular stage of the system.

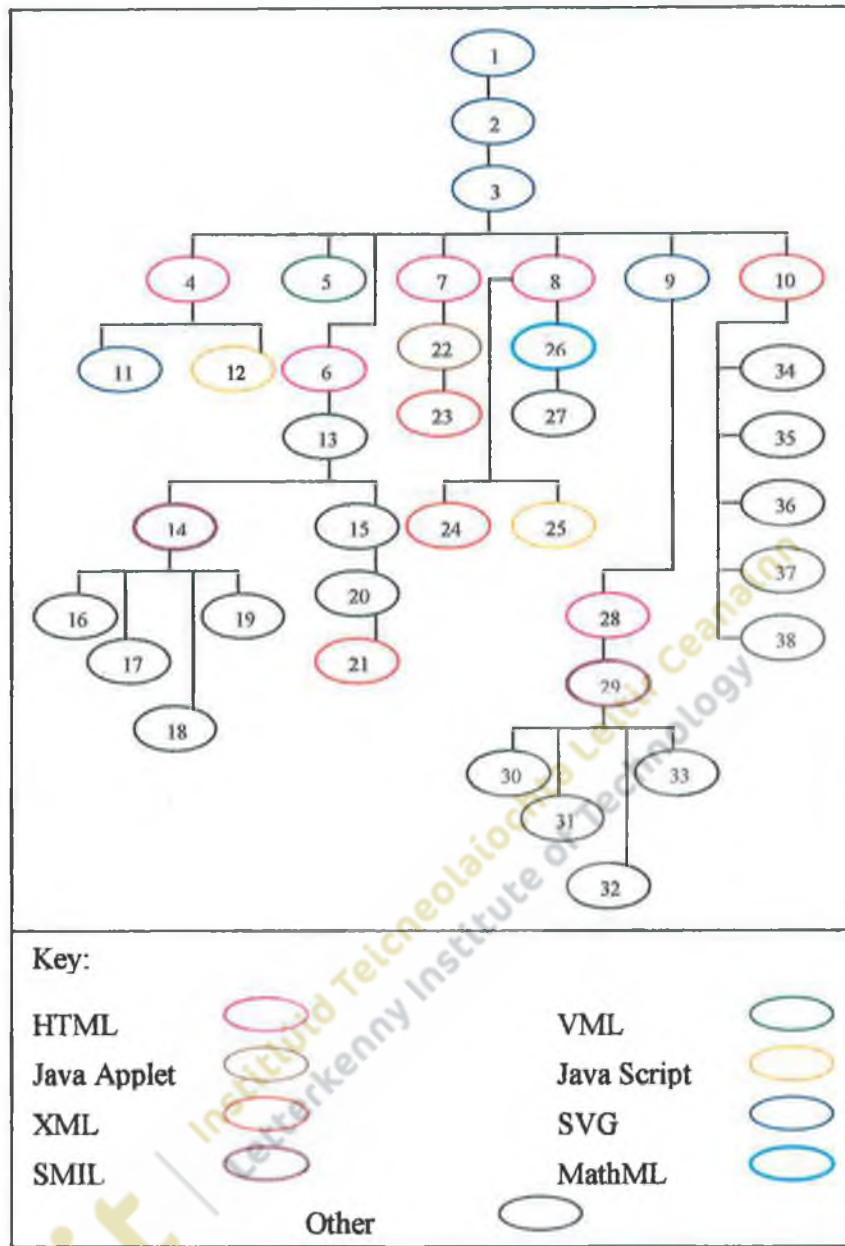


Fig. 3.10 System Structure Diagram

As set out in Chapter 1, it was necessary to determine the interoperability of XML with other development languages. Figure 3.10 shows the various languages as they interact with each other. The development of the systems showed how XML is very interoperable and can be easily used with other languages such as Java. Appendix G shows print screens of the systems running and the various languages interacting.

Node No.	Language	Explanation
1, 2, 3	SVG	Produces the graphics and links on the opening screens/"home page"
4	HTML	Used to embed SVG within
5	VML	Used to create the help pages. Uses HTML as a base
6, 7, 8	HTML	Used to access the database and to embed Applets/MathML within
9	SVG	Used to provide a link to the SMIL presentation
10	XML	Formatted and validated to display data in various forms
11	SVG	Animation
12	JavaScript	Used to animate and create 'onmouseover' events, etc.
13	Tomcat	Application Server, used to retrieve data from MS Access
14	SMIL	Synchronises the various media elements incorporated
15	JDBC	A Thin Driver used to access the data in the database
16	AVI	Video Format (used in SMIL presentation)
17	Real Text	Text document (used in SMIL presentation)
18	AU	Audio format (used in SMIL presentation)
19	GIF	Image format (used in SMIL presentation)
20	MS Access	Used as a relational database to retrieve data from
21	XML	The data from the database is displayed in XML format
22	Java Applet	Used to move through an XML file to retrieve Book information
23, 24	XML	The XML is acting as a 'database' from which data can be accessed
25	Java Script	Used to access elements in the DOM
26	MathML	Displays mathematical notations in the browser
27	XSL	Used to 'transform' the XML data for presentation purposes
28	HTML	Used in order to 'embed' the SMIL presentation within
29	SMIL	Synchronises the audio, video and text in a markup language
30	AVI	Video Format (used in SMIL presentation)
31	AU	Audio Format (used in SMIL presentation)
32	Real Text	Text document (used in SMIL presentation)
33	GIF	Image Format (used in SMIL presentation)
34	CSS	Cascading Style Sheet, used to format the XML for display (colour, etc.)
35	XSL	Extensible Stylesheet Language, used to create scrollbars, etc.
36	XML Schema	Validates the XML document (similar to the DTD)
37	XML Namespace	Ensures that the tags are unique
38	DTD	Document Type Definition, used to create a 'valid XML document'

Table 3.1 Key to Figure 3.10

3.6.7 Jackson System Development

A JSD can be considered as a 'second-generation' software design method, as it was developed during the late 1970s and early 1980s [ERDB]. As a method, it encompasses both analysis and design activities. It is these activities that help a designer construct a model of a system in terms of a set of 'long-running' interacting concurrent processes of its description.

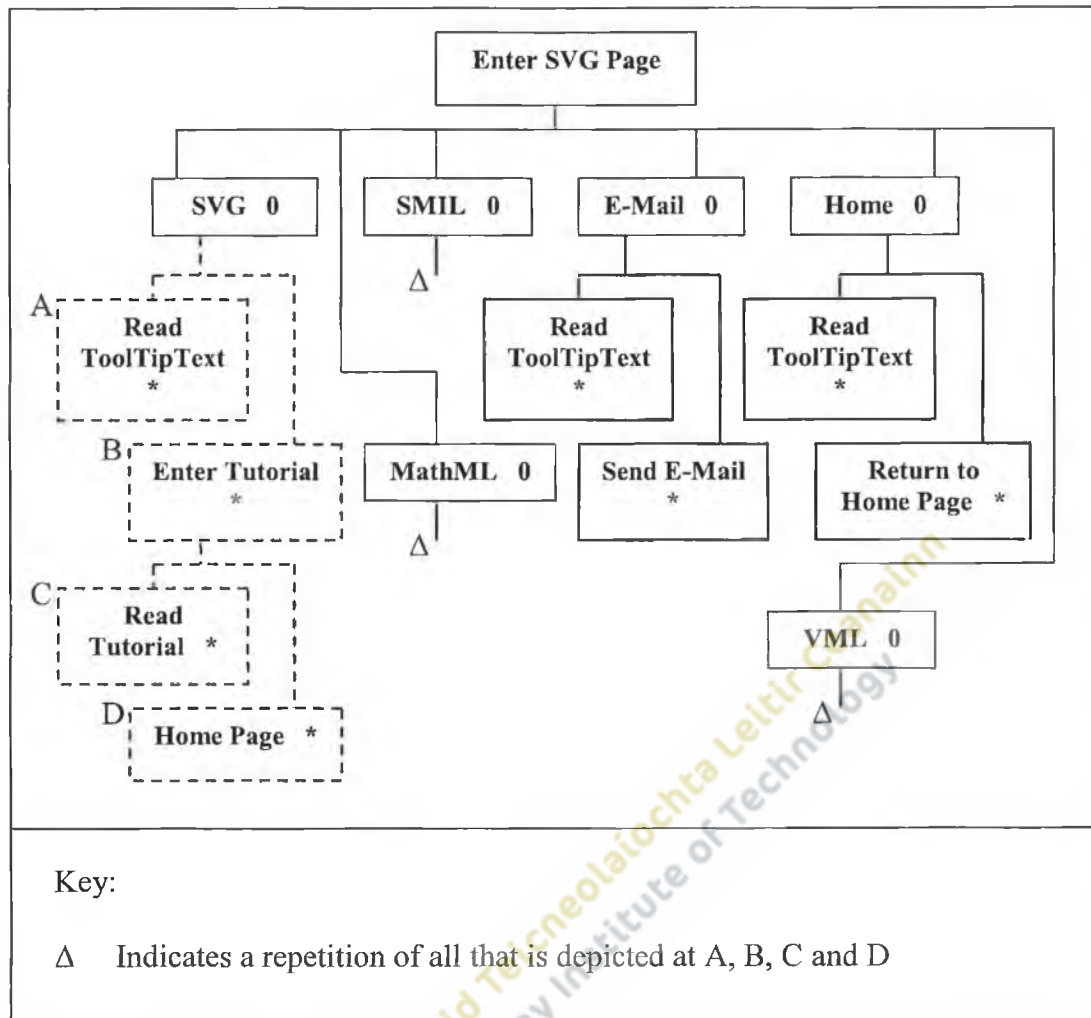


Fig. 3.11 JSD⁵ Diagram depicting the main operations within the SVG Page

Figure 3.11 shows how the main functions within the SVG Page interact. It is possible for the user to select a tutorial on any of the topics, look at the support available, send an E-mail or go back to the home page. It can be seen that from each tutorial it is possible to: Read ToolTip text, Enter Tutorial, Read Tutorial and return to the Home Page repetitively. This is further depicted in the screen shot shown in Figure 3.12:

⁵ Jackson System Development

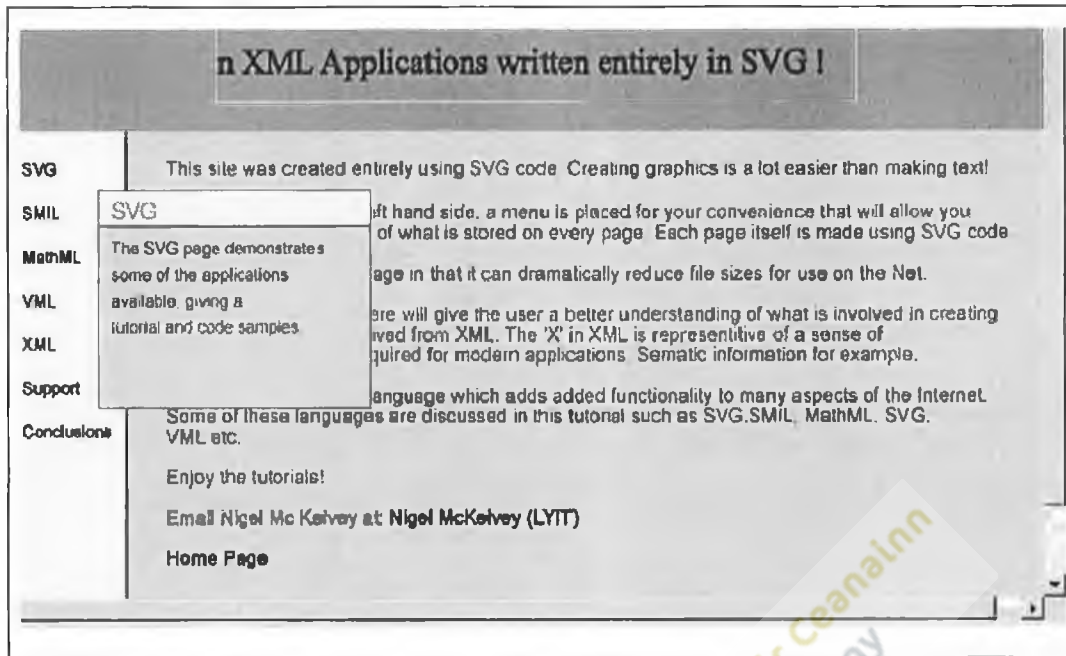


Fig. 3.12 Screen shot of the 'SVG Page'

3.6.8 Entity Relationship Diagrams

An Entity Relationship Diagram (ERD) is primarily used to identify the relationships that exist between static data objects in a problem model or in a design model [ERDB]. These diagrams can be used to model the detailed data stored in a system and they can also be used to model the relationships that exist between more complex and abstract design 'objects'. Several types of relationships may connect entities. As well as this, attributes can be composite, with higher-level attributes being decomposed into lower-level attributes [ERDB]. Relationships are categorised by their 'n-ary' properties, which include 'one-to-one', 'one-to-many' and 'many-to-many' [ERDB]. ERDs can be used during the analysis stage of the problem and also during the development of the solution.

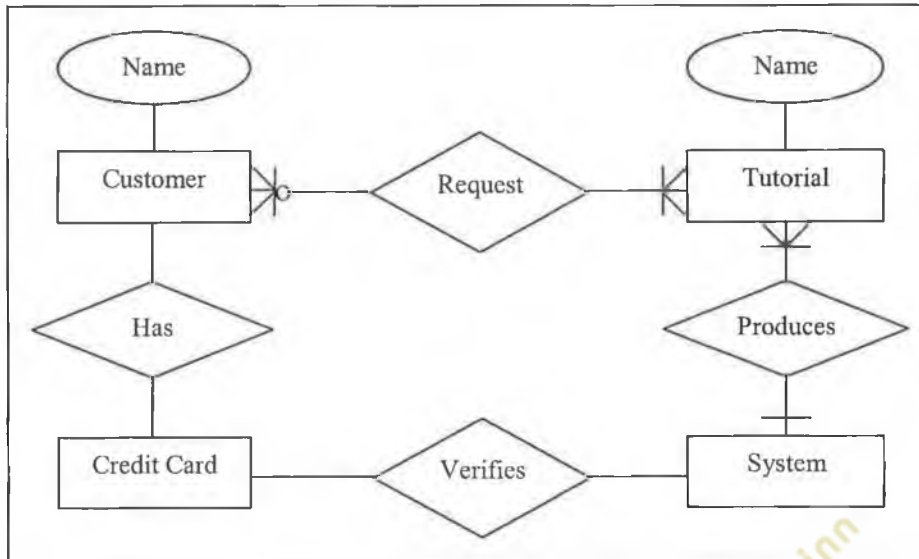


Fig. 3.13 ERD – Ordering a Tutorial

Figure 3.13 shows how the entity Customer has an attribute 'Name' and can order many Tutorials. A Tutorial also has an attribute 'Name'. It can also be seen that the System produces many Tutorials and the System also verifies Credit Card details, which the Customer might have.

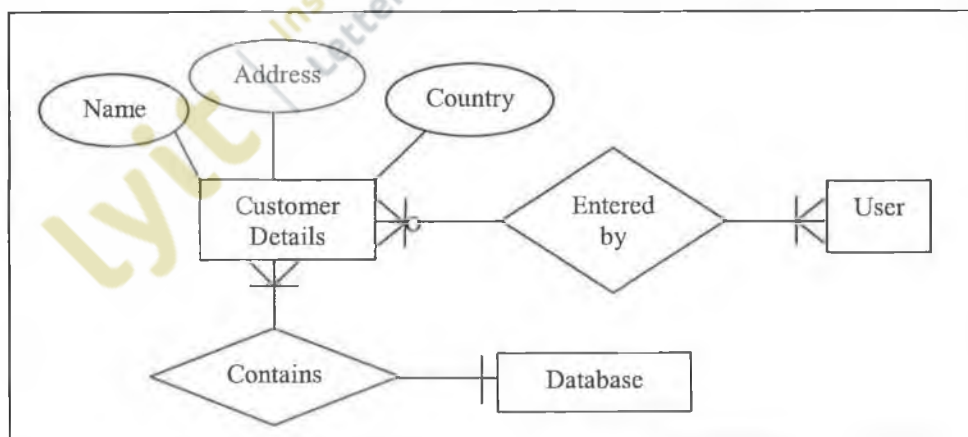


Fig. 3.14 ERD – Entering Customer Details to the Database

Fig 3.14 is an ERD diagram that depicts how Customer Details are entered into a database. Customer Details (for demonstration purposes) has three attributes: Name, Address and Country. Many Users can enter many Customer Details. It can also be seen that many Customer Details are contained in one Database.

3.7 Relational Databases

Databases are essentially specialised tools for data storage [JDBR]. File systems are good for storing and retrieving a single volume of information associated with a single virtual location. Databases provide applications with an established data storage and retrieval system based on mathematical theories about data devised by Dr. E. F. Codd [JDBR]. Conceptually, a relational database can be pictured as a set of spreadsheets in which rows from one spreadsheet can be related to rows from another. In reality however, the theory behind databases is much more complex. Each 'spreadsheet' in a database is called a table. As with a spreadsheet, a table is made up of rows and columns as can be seen in the example given Table 3.2.

ISBN	Title	Year	Author
0-45678-9654	XML Book 1	2002	Joe Bloggs
1-87523-77-00	Database Book 2	2003	John Doe

Table 3.2 Book Details represented in a table

XML is a good language for data exchange and it is frequently used in communications between systems [DXDB]. Complex querying and information retrieval requires storing metadata in a database. Contemporary database technologies, such as Oracle, are expanding to model XML data representations and

new native XML database technologies are also emerging [MRKM]. Tamino⁶ and XYZFind⁷ are two examples of native XML databases [MRKM]. However, loading XML documents into existing relational databases can be problematic. One of the main challenges is to map the semantics of the XML into appropriate semantics for the relations [DXDB]. This difficulty needed careful consideration when designing the database system for the application.

3.7.1 Advantages of Relational Databases

Data entry, updates and deletions are efficient.

Data retrieval, summarisation, and reporting are efficient.

Database behaves predictably.

Changes to the database schema are easy to make.

3.7.2 Disadvantages of Relational Databases

The user must define all the database relationships up front.

Users must know what the key elements are in order to access data from multiple databases.

3.8 Java and XML

The Java platform is Internet-enabled, which assists connectivity over TCP/IP⁸ between exchanging parties [XJAV]. As a result, such parties can employ the Internet as an exchange transport. Information can be exchanged from one organisation to

⁶ Product of Software AG

⁷ XYZFind Corporation is a pioneer in the field of XML database technologies. Its product, XYZFind Server, is a native XML database.

⁸ Transfer Control Protocol/Internet Protocol

another in a 'tree-like format' or as formatted information. Additionally, technically sophisticated enterprises can utilise tools and technologies available to help smaller enterprises participate in electronic data exchange.

Both XML and the Java platform inherently support Unicode character sets and as a result both environments enable enterprises to sustain the development of internationalised applications [XJAV]. By incorporating the Unicode standard, applications can represent characters in multiple national languages. With a combination of XML markup as the format for data exchange and an internationalised application written in Java, then XML documents can be exchanged globally.

When deciding which programming language to combine XML with, it was deemed viable to incorporate Java for the aforementioned reasons as its appraisal has been widely documented and its parsing capabilities widely used. There is also a large range of middleware (APIs) written in Java for managing data either in XML or with XML input or output.

3.9 Boundary Conditions

As a result of XML being quite new, detailed consideration was given to the selection of a browser that would offer the best support, and which additional software/plugins and/or hardware would be required (refer to Chapter 4). As a result, the research carried out outlined what could and could not be accomplished within the time frame. The following were considered to be within the boundary conditions:

3.9.1 Within Boundary Conditions

Table 3.3 outlines the languages that could be incorporated successfully in the first application (“Sigma-X”), which would run on MS Internet Explorer 5.5. refer to Chapter 4 for further technical considerations.

Language	Demonstrates:
SVG	“Home Page”, Animation, Flash Alternative, Tutorials, Graphics, Links, Menus, Pan/Zoom Capabilities, Facilitates Disabilities, File size reduction, Text Displays, Alternative to VML
VML	“Help” Pages, Text displays, Alternative to SVG, File size reduction, Links
SMIL	Combining media elements, CBT potential, Facilitates numerous disabilities, Streaming media elements
MathML	Potential when combined with XSL, Uses to mathematicians, Support available for the language
XML	Display of text, Tutorials, Links, Menus, Formatting with XSL, Formatting with CSS, Using XML Schemas, DTDs, Namespaces, Data Islands, Using JavaScript, Comparison with VML and SVG tutorials, XML as a database, Accessing the DOM, Representing legacy data as XML from MS Access
Java	Retrieving data from a relational database, Formats the data as XML, Interoperability, Platform Independence, Unicode aware
JavaScript	Animation potential when combined with SVG, Calculations on data, Accessing the DOM

Table 3.3 Sigma-X: Tutorial Demonstration/Applications of XML

Table 3.4 outlines the languages that were used together in order to upload XML documents into a relational database in Oracle8i. It runs on Internet Explorer 5.5. The second application was necessary as permissions set on the machine that ran “Sigma-X” would not allow Oracle to be configured in such a way that the developer could bypass telnet (refer Section 3.6.3.2).

Language	Demonstrates:
SVG	Textual descriptions, Links, Animation
Java	Uploads XML documents into Oracle8i, Parsing, Interoperability, Platform Independence, Unicode aware
SQL	Manipulates the data in the database, Executes queries
Java Applets	GUI, Data manipulation in the database
XML	XML's hierarchy, How legacy data can be manipulated

Table 3.4 Omikron-B: Uploading XML into Oracle8i

Table 3.5 describes how a VoiceXML application was created. The application required the use of a piece of software called the IBM WebSphere Voice Toolkit and an SDK server on which to run (refer to Chapter 4).

Language	Demonstrates:
VoiceXML	Voice Commands, Speech Synthesis, Text-to-Speech technology, CBT Potential, facilitates certain disabilities, information retrieval, grammar and pronunciation creation
Java Script	Credit Card verification

Table 3.5 Kappa-C: VoiceXML Capabilities

3.9.2 Outside Boundary Conditions

Certain restrictions were evident that would limit the development of the applications and would also restrict how users would interact with the system. They are listed as follows:

“Omikron-B” and “Sigma-X” could not be combined into one application, as network permissions on the machine would not allow Oracle to be configured as required. Users were limited to how they could access data in the database in Oracle8i. This is because (as a design consideration) constraints were put on the tables.

Developing a VoiceXML application that could be incorporated into one of the other applications would require an extensive amount of time in order to achieve an acceptable application.

It was outside the boundary conditions to run the same applications on Netscape 6.2 and/or Opera as support for the various applications were limited on these browsers.

Research also revealed that a more 'interactive' tutorial would not be as viable as first anticipated as it would require additional JavaScript that was deemed outside the scope of the proposed project.

3.10 Conclusions

Relevant aspects of software design have been considered in this chapter and this section summarises some of the issues encountered such as, interface considerations, database connections, configurations and browser support, etc.

Designing the interface whilst considering certain HCI factors proved challenging. It was necessary to design an interface that was aesthetically pleasing as well as practical and also incorporating the various different applications of XML that were being studied, e.g. SVG, VML, SMIL, etc.

This thesis has been compiled to investigate the interactions of XML with databases. It is therefore necessary to strive to amalgamate all of the technologies into one seamless application. After extensive research, it was concluded that Java would be utilised in conjunction with XML in order to connect to two databases set up in both MS Access and Oracle 8i. Connections to them via JDBC and a thin driver respectively were also required.

In order to connect to MS Access the DSN had to be configured in order to specify the table within the database to be accessed. The selection of the database allows the system developer to specify one database that the system will communicate with, or a number of databases that allows the system users to access data stored in several tables, using the one driver. Thus, a connection may be established with the client.

Again, access to tables contained within Oracle required a client side SQLPlus connection to be set up directly without using Telnet. In order to do this it was necessary to install Oracle Enterprise Manager and its client side version of SQLNet on the PC.

Blending all of the XML applications studied into one system proved quite difficult. Support for the various markup languages was very limited and varied (refer to Chapter 5, Section 5.7, Table 5.2). A browser was required that would support the following: Java Applets, Java Script, Frames, XML, XML Schemas, DTD, Data Islands, SVG, VML, MathML and SMIL. Various browsers were considered such as Opera 6.0 and Netscape 6.2. However, anomalies arose especially when trying to display various multimedia elements through SMIL or displaying mathematical notations with MathML. Plugins were embedded within the code in order to display/utilise the required functionality within the system (refer to Chapter 2 Sections 2.13 (Table 2.5) and 2.14 (Table 2.6) and Chapter 5, Section 5.7 (Table 5.2)). Therefore, it was concluded that Internet Explorer 5.5 was a more acceptable option as all of the required XML applications could be displayed by incorporating various plugins outlined in Table 5.2 of Chapter 5.

Kappa-C however, could not be embedded within a Web page. The capabilities of VoiceXML were then demonstrated in a separate application (Kappa-C), which required the IBM WebSphere Voice Toolkit to be installed on the machine as well as the IBM WebSphere Voice Server for Windows 2000 and the Java 2 SDK Standard Edition. The user could then interact with the application using a microphone and/or the keyboard. From the research conducted, it became apparent that it is becoming increasingly important for Web designers to cater for the needs of those with disabilities. VoiceXML could prove to be very influential in the future of Web development as it eliminates the necessity for the user to have to look at the Web site at all. The application could relay all required information to the user and the user could in return respond to only the desired information. The capabilities of SALT (refer to Chapter 2, Section 2.16) will enable 'voice' applications to be embedded within Web pages.

The applications developed demonstrate the capabilities of XML and also indicate how this relatively new language can interact with Legacy Databases. The applications reveal how XML can be utilised to bring about Web advancements that will improve networking capabilities. XML will further advance in the years to come and so it would be unwise of software houses to neglect facilitating support for these applications to run in their browsers.

In order for these applications to be successfully developed, several hardware and software issues needed careful consideration. Many of XML's application can not operate without the incorporation of certain plug-ins for example. This is discussed

further in the next chapter, which includes summaries of the technologies (i.e. software/hardware requirements) that were used when developing the applications.



Chapter 4

Technologies Employed

lyit | Institiúid Teicneolaíochta Leitir Ceanaínn
Letterkenny Institute of Technology

4.1 Introduction

It is necessary at this juncture to summarise the technologies implemented and incorporated in the applications developed. This chapter will document each of the technologies, outlining the availability of support by browsers for them and where they can be obtained as well as specifying any software/hardware and/or plugins that were required in order to develop the applications. The following will be discussed with relevance to the aforementioned points:

XML	SVG
VoiceXML	Java
VML	JDBC
SMIL	Oracle8i
MathML	MS Access

Table 4.1 Technologies Implemented

4.2 Required Software/Hardware

The following sections document the hardware and software requirements for the technologies mentioned in section 4.1 above:

4.2.1 SVG Requirements

SVG proved to be a very interesting means of presenting graphics. Certain requirements however needed to be in place first.

Name	Size	Description	Cost	Provider
Adobe SVG Viewer 3.0	4.66 MB	A plugin that enables a browser to display SVG files successfully.	Freely Downloadable from www.apple.com	Adobe
Amaya	9.98 MB	WYSIWYG interface used to develop applications for the Web.	Freely Downloadable from www.w3.org	W3C
Internet Explorer 5.5	18.4 MB	A browser used for displaying Web pages.	Freely Downloadable from www.microsoft.com	Microsoft

Table 4.2 SVG Requirements

The Adobe SVG Viewer 3.0 had to be installed, which is a plugin that allows SVG code to be recognised by the browser and displayed to the screen. It essentially means that the .svg extension is recognised by the browser (in this case MS Internet Explorer 5.5). The graphics could be created in a normal text editor if desired but Amaya allows a developer to use a GUI-driven tool to create graphics using the mouse and specifying the attributes (stroke width, colour, font, etc.) through menu options.

4.2.2 VoiceXML Requirements

Most of the technologies implemented when developing the applications required additional software of some kind. However, VoiceXML required more than others. The IBM WebSphere Voice Toolkit enables a developer to create VoiceXML applications using an integrated environment. The software can run on the desktop and provides a GUI-driven tool for facilitating application development.

Name	Size	Description	Cost	Provider
IBM WebSphere Voice Server for Windows 2000 (SDK)	53.6 MB	Uses VoiceXML technology to enable developers to create voice-enabled applications and test them on a desktop workstation.	Freely Downloadable From www-3.ibm.com	IBM
IBM WebSphere Voice Toolkit	425 MB	Helps to create VoiceXML applications using an integrated development environment.	Freely Downloadable From www-3.ibm.com	IBM
Java 2 Runtime Environment Standard Edition v1.3.1	22.1 MB	It provides software developers with a platform for rapid application development.	Freely Downloadable from http://java.sun.com	Java
Java 2 SDK Standard Edition v1.3.0_02	288 MB	Includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.	Freely Downloadable from http://java.sun.com	Java
Plantronics Stereo PC Headset	N/A	A device with headphones and a microphone.	€118 (Euro) Approx.	VC Computer, Ireland

Table 4.3 VoiceXML Requirements

However, this software would not execute a VoiceXML program correctly unless the IBM WebSphere Voice Server for Windows 2000 (SDK) was in place. The SDK Server provides the following capabilities:

- VoiceXML Speech browser,
- IBM speech recognition engine,
- IBM TTS engine and
- Tools for developing/testing VoiceXML applications.

This SDK Server has certain system requirements also. It needs a 366 MHz processor, 128 MB RAM, 290 MB disk space, a display adapter setting of greater than 256 colours and a Windows 2000 compatible 16-bit, full-duplex sound card. As Table 4.3

shows, VoiceXML also required the Java 2 Runtime Environment Standard Edition to be in place and a microphone/headset for communicating with the system developed.

4.2.3 SMIL Requirements

It proved difficult to incorporate SMIL as the correct combination of media elements needed to be employed that would be compatible with the Player used.

Name	Size	Description	Cost	Provider
RealOne Player	22 MB	Facilitates audio and video playing on the Web	Freely Downloadable from http://service.ral.com	RealNetworks
Internet Explorer 5.5	18.4 MB	A browser used for displaying Web pages.	Freely Downloadable from www.microsoft.com	Microsoft
MGI Videowave	36 MB Approx.	Allows a developer to retrieve video from a camcorder.	\$49.95 (US Dollars) Approx.	MGI
Panasonic NV-DS150 Digital Camcorder	N/A	A camcorder required for shooting the videos.	£789.99 (UK pounds) Approx.	Payless Electrical, UK

Table 4.4 SMIL Requirements

For the application developed, the RealOne Player (G2) was used. Table 2.5 – “Media support in SMIL”, in Chapter 2, outlines which players support which media elements. This was a very important aspect that required careful consideration when designing the application (this is further discussed in Chapter 3 and Chapter 5). In order to display the SMIL presentation, it was necessary to ‘embed’ the presentation within a HTML document as can be seen in Code Listing 4.1. The embedded presentation was displayed using Internet Explorer 5.5.

```

<object id="media"
  classid="CLSID:CFCDA03-8BE4-11CF-B84B-0020AFBCCFA" width="550"
  height="410">
  <param name="src" value="SMIL_Presentation_Name.smil">
  <param name="console" value="Clip1">
  <param name="controls" value="ImageWindow">
  <embed controls="ImageWindow" console="Clip1"
  type="audio/x-pn-realaudio-plugin"
  src=" SMIL_Presentation_Name.smil"
  width="250" height="190" autostart="true">
  </embed>
</object>

```

Code Listing 4.1 Embedding a SMIL presentation into an HTML Page

4.2.4 VML Requirements

VML required very little additional software or hardware to be installed in order to run applications. The VML Generator is a useful tool when writing VML but it is not essential. It does however save the developer a lot of time when creating graphics. The main requirements that the VML Generator needed before it could be installed is that the Operating System be Windows NT, 2000 or 9X, the browser used must be MS Internet Explorer 5.0 or later, 166MHz processor, hard-disk space of 7MB and a video resolution of 800 x 600 or higher.

Name	Size	Description	Cost	Provider
Microsoft VML Generator	868 KB	A piece of software that allows developers to create VML graphics using a GUI.	Freely Downloadable from http://msdn.microsoft.com	Microsoft
Internet Explorer 5.5	18.4 MB	A browser used for displaying Web pages.	Freely Downloadable from www.microsoft.com	Microsoft

Table 4.5 VML Requirements

4.2.5 MathML Requirements

A MathML file created using Amaya, can be saved with the extension .mml and rendered in Internet Explorer 5.5 provided the IBM Techexplorer plugin is installed.

But it is also possible to render MathML in Internet Explorer 5.5 using XHTML (refer to Section 2.13, Chapter 2).

Name	Size	Description	Cost	Provider
Amaya	9.98MB	A WYSIWYG interface for developing Web applications.	Freely Downloadable from www.w3.org	W3C
Internet Explorer 5.5	18.4MB	A browser used for displaying Web pages.	Freely Downloadable from www.microsoft.com	Microsoft
IBM Techexplorer Hypermedia Browser ¹	17.7MB	A Plugin that will allow a .mml file to be rendered in a browser.	Freely Downloadable from http://www-3.ibm.com/software/network/techexplorer/	IBM

Table 4.6 MathML Requirements

4.2.6 Additional Requirements

Table 4.7 shows some additional browsers that were downloaded for comparison purposes. It was necessary to outline how varying browsers would render applications of XML. This revealed some interesting facts as it showed how an application might be rendered well in Internet Explorer 5.5 and not in Netscape 6.2 and vice versa. Other pieces of software were also required for the development of the SMIL applications for research purposes.

Name	Size	Description	Cost	Provider
Netscape 6.2.1		Browser	Freely Downloadable from http://wp.netscape.com/browsers/6/	Netscape
SMIL 2.0 Player	4.50MB	Allows SMIL presentations to be viewed.	Freely Downloadable from www.inobject.com	InterObject
QuickTime	9.96MB	Allows a developer to play audio and video media elements.	Freely Downloadable from www.apple.com/quicktime/download/	QuickTime
Opera 6.0 (Win32)	12.7 MB	Browser	Freely Downloadable from http://download.com.com	Opera

Table 4.7 Additional Requirements

¹ Introductory Edition

4.3 Technologies Employed

The following technologies were employed in “Omikron-B”:

4.3.1 Oracle8i

Oracle8i is a client/server database. This implies that the database server runs independently from the applications that may be used to access it [ORLG]. There are four basic steps involved with the client/server architecture in Oracle8i [ORLG]:

1. The server listens for requests,
2. It accepts requests from clients,
3. The requests are processed and
4. The results are sent back to the clients.

The Internet is becoming more influential in the day-to-day running of a modern business and being able to interact electronically with the customers is rapidly becoming a requirement within any Web site. It was for this reason that a three-tier client/server architecture was required.

4.3.1.1 Three-tier Client/Server Architecture

A three-tier Oracle application still has an Oracle server and client PC. However, an application server sits between the PC and database. A lot of Internet companies already have Oracle as a basis for managing their information for the Web, which is why XML could play an important part when interacting with an Oracle database.

This concept is further discussed in Chapter 2 Sections 2.17 and 2.18.

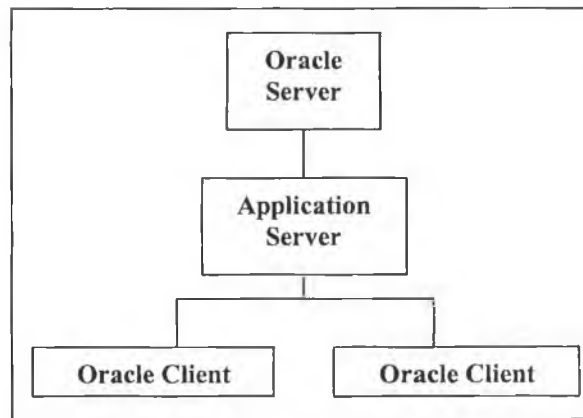


Fig. 4.1 Three-tier Client/Server Architecture

This connectivity was established as a result of incorporating a thin JDBC driver. The JDBC Thin driver uses Java to connect directly to the Oracle Server and is typically used for Java applets in either a two-tier or three-tier configuration, though it can also be used for Java applications. The JDBC Thin driver provides its own implementation of a TCP/IP version of Oracle's SQLNet. A JDBC Thin driver is written entirely in Java, and as a result the driver is platform-independent.

4.3.1.2 Advantages of Three-tier Client/Server

- **Scalability:** Allows the developer to spread clients over a number of application servers.
- **Ease of Distribution:** Allows the developer to distribute the software enabling anyone using a standard Web browser to access it [ORLG].

4.3.1.3 Disadvantages of Three-tier Client/Server

- Creates an increased need for network traffic management, server load balancing, and fault tolerance.

4.3.2 Microsoft Access 2000

Microsoft Access was chosen because it is widely used in small businesses. Its potential when interacting with XML required an investigation in order to determine how viable it would be to incorporate this database system when communicating data across an intranet for example.

Before designing a database, it is necessary to decide what information should be collected, how that information will be used, and who will be using it. If these precautions are not taken then the database has the potential to be disorganised and possibly unsuitable to its intended purpose. The information stored is valuable, however, if this data (legacy data) can be retrieved and displayed in another format, then its value is increased even further. Research conducted revealed that developers could create sophisticated enterprise-wide database solutions that integrate easily with the Web. Connections to the database were via the JDBC-ODBC Bridge (refer to Section 4.3.2.1).

4.3.2.1 Creating a Data Source with ODBC

When creating a connection to a data source using ODBC, the establishment of a Data Source is essential. The development of this data source may be seen in the following steps:

- Upon selection of the Control Panel, the system developer is required to select the 32 bit Windows.
- A dialogue then appears known as the Data Source Administrator (DSA) (see Figure 4.2). The System DSA is then selected.

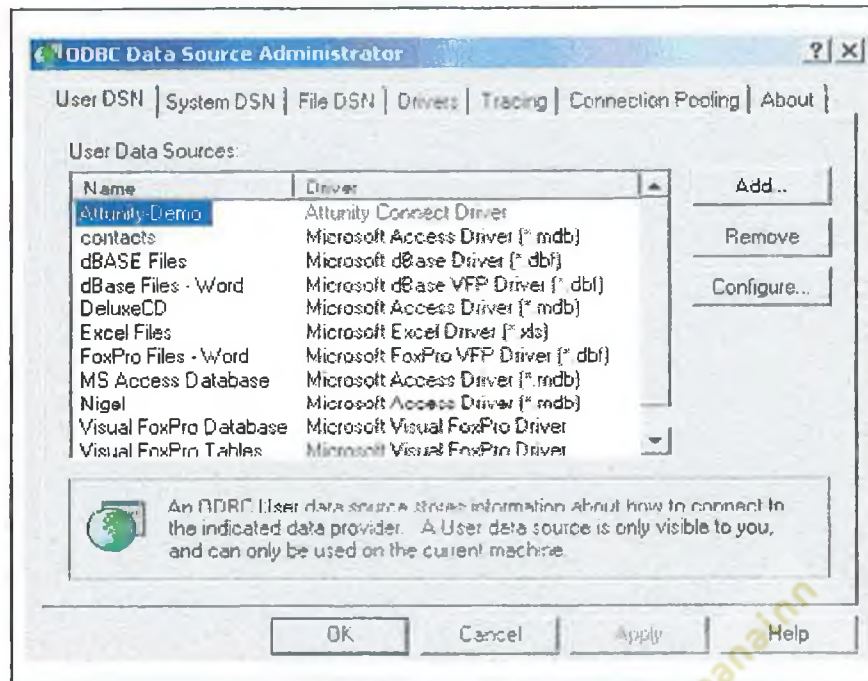


Fig. 4.2 Data Source Administrator Options

By clicking on the Add button, the System DSA then provides a list of drivers.

Microsoft provides this list of drivers as shown in Figure 4.3:



Fig. 4.3 Microsoft Drivers

It was then necessary to click on the Microsoft Access driver option and click on Finish. The dialog box in Figure 4.4 then appears:

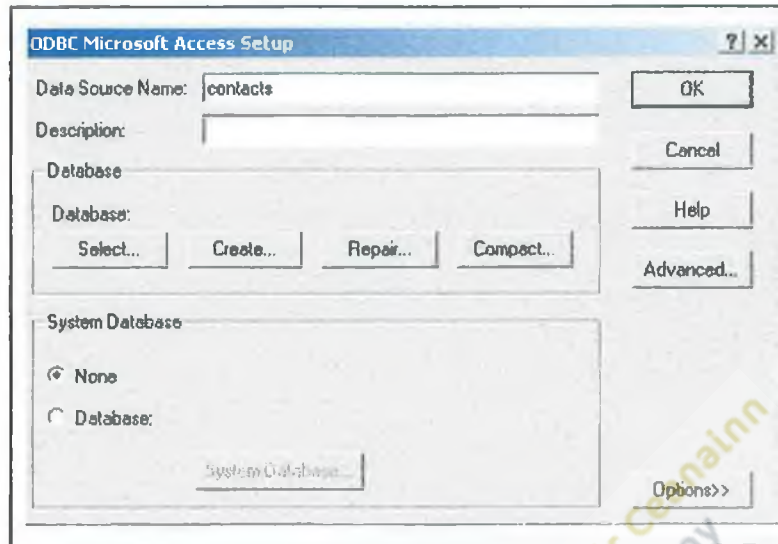


Fig. 4.4 Entering the DSN

In order to get a connection to the database, it was necessary to type in the name of the database in the Data Source Name text field. When this was completed it was essential to click on the Select option and browse to the location of the *contacts.mdb* file on the machine. When this was successfully completed the original dialog box reappeared with the name of the database present. This implies that it is then possible to execute a query on the database from the application developed. It is very important that a Data Source Name be entered. This is the name through which the data Store is referred to throughout the system. The Selection of the database allows the system developer to specify one database that the system will communicate with, or a number of databases that allows the system users to access data stored in several tables, using the one driver. Thus, a connection may be established with the client.

A series of Java programs were written that would take an SQL command entered by the user and connect to the required database using the correct URL, username and

password, etc. via the JDBC-ODBC Bridge. Figure 4.5 depicts how the SQL command can be taken and executed:

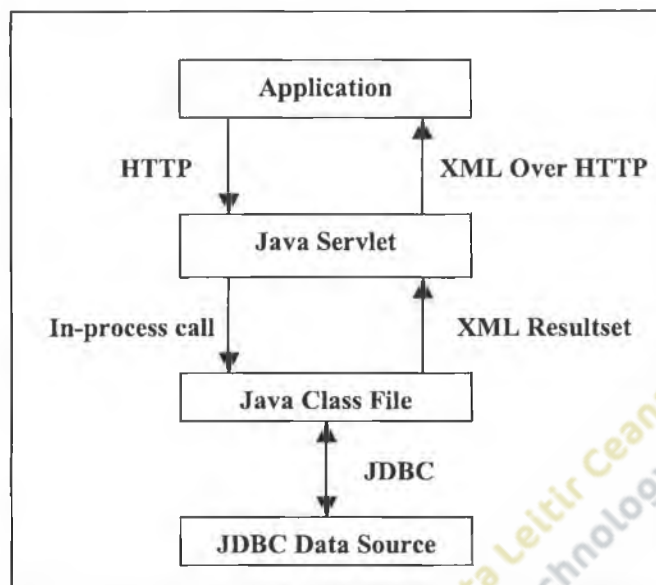


Fig. 4.5 Servlet handling a request

4.3.3 JDBC

JDBC is an API that allows a user to access most data sources from the Java programming language. An SQL-Level API means that JDBC facilitates the construction of SQL statements and embeds them inside Java API calls [JDBR]. JDBC allows the user to translate efficiently between the database and the Java application. The results from the database are returned as Java variables and access problems are thrown as exceptions. JDBC and XML can be used together, as they help to implement data management [ACMX].

The JDBC API defines Java classes to represent database connections, SQL statements, result sets, database metadata, etc. It allows a Java programmer to issue SQL statements and process the results. JDBC is the primary API for database access

in Java. The JDBC Driver API represents the set of interface classes, any new JDBC compliant driver should implement. This reveals inherent extendibility for different types of databases. The remote access to a database over the network depends upon the particular installation of the JDBC driver for that database.

By XML-enabling a JDBC application it helps promote platform-neutral and device-independent access to data [PDBW]. By incorporating XML, XSLT and JDBC it is possible to design scalable and extensible architectures that assist device-independent data delivery [PDBW]. This enables end users to access the data in a Web browser or through other devices such as PDAs.

When designing the prototype for this system, it was deemed necessary to have two applications accessing data in two different databases (refer to Chapter 3 Section 3.9). The two chosen databases were Oracle 8i via the Oracle thin driver and Microsoft Access via the Java Database Connectivity – Open Database Connectivity (JDBC-ODBC) Bridge. The use of JDBC as the data access API allows the user to have access to legacy relational data.

4.3.3.1 Java Database Connectivity – Open Database Connectivity (JDBC- ODBC) Bridge.

Due to the implementation of JDBC through Java, the data is portable and secure. As a result, the calls to SQL commands are acceptable through Applets if the connection is Java. ODBC, on the other hand, is not feasible in Applets due to security restrictions.

The JDBC-ODBC Bridge enables communications to a database through the translation of JDBC statements to/from ODBC statements. Statements issued in Java seeking connection to a database may not have direct connection to that database and statements implemented by ODBC might not be issued in Java. As a result a bridge or 'translation' is required. The translation is implemented using a Java package known as 'jdbc.odbc'. Within this package a library exists which is used to access ODBC [JDBR].

4.3.3.2 Advantages of JDBC-ODBC

- It can establish a connection with a database.
- It can send SQL Statements
- It can process the results.
- Simple to develop.
- Bridges such as JDBC-ODBC allow JDBC to access almost all databases.

4.3.3.3 Disadvantages of JDBC-ODBC

- It is possible to fetch a series of table rows in a forward direction, but it is not possible to move backward in a table to rows that have been previously fetched.
- Reissuing a query just to reload a row may cause delays.
- Security Restrictions.

4.3.4 Tomcat Version 3.2

Tomcat is a servlet container and it is a means for implementing JavaServer Pages. It may be used in stand-alone mode, or in conjunction with several web servers that are in existence, such as:

- Apache, version 1.3 or later
- Microsoft Internet Information Server, version 4.0 or later
- Microsoft Personal Web Server, version 4.0 or later
- Netscape Enterprise Server, version 3.0 or later

Tomcat is itself a well-developed HTTP server, meaning that tests can be conducted without having to configure other server technologies [TMCL]. It is possible to check if Tomcat is operational by pointing the browser at port 8080 on the computer where it has been started (i.e. `http://localhost:8080` or `http://127.0.0.1:8080`). Tomcat is a high performance servlet container that is relatively easy to use and includes a HTTP listener that eliminates the necessity to install a separate Web server [PDBW].

By using Java/JSP some important factors had to be considered when designing the application such as security issues. The combination of the two provides flexibility, portability, ease of development, and an availability of a rich set of APIs [NWMG].

4.4 Alternatives to Java

Tcl/tk could be considered as an alternative to Java. Tcl/tk could be advantageous as it is easily extendible, easy to integrate with C and is relatively easy to combine with a variety of packages [DXDB]. Nevertheless, it is not as well known as Java or JavaScript. Tcl/tk is also an excellent prototyping language because it is an untyped scripting language, which incorporates flexible data structures and comparatively

expensive built-in features [DXDB]. However, certain disadvantages exist with Tcl/tk in that, it has a small user community and there tends to be less support and maintenance of public packages and a slower response to integrating new technologies [DXDB]. For these reasons Tcl/tk was not further researched for this thesis.

4.5 Conclusions

Incorporating XML technologies into an application may provide several advantages (e.g. reduced file sizes, additional metadata, etc.), however, getting the application to the point where it can run successfully can be problematic. Writing SVG, VML, SMIL, MathML or VoiceXML programs is not as simple as it may appear. Several pieces of software/hardware and plug-ins need to be in place before the application will run successfully. In the case of SMIL and VoiceXML applications this can incur additional costs for a businesses as additional hardware and/or software may be required. An advantage with these technologies is that most of the software required is freely downloadable from the Internet.

In order to connect to a database such as Oracle8i or MS Access a Java Development Kit (JDK) must be installed. As discussed in Chapter 2 Section 2.17, linking to a database is quite difficult and with license agreements, can be quite costly for a small business.

With the technologies in place and the prototype applications developed (for screenshots, refer to Appendix G), it is now necessary to test the functionality and the capabilities of XML as discussed in previous chapters.

Chapter 5

Testing and Evaluation

lyit

Institiúid Teicneolaíochta Leitir Ceanaínn
Letterkenny Institute of Technology

5.1 Introduction

This chapter tests and evaluates the applications as described in the previous chapter and in Appendix G, based on users' responses to a 'Questionnaire' and 'Assignment Sheet' (refer to Appendix F). Issues regarding the level of support available for XML, the interaction of XML with legacy databases and the portability of XML are discussed. The results obtained enabled a set of conclusions to be drawn based on the applications developed.

5.2 Introduction to Testing Methodologies

One of the most important stages in the development of a piece of software is the testing phase. This phase may take as long as the development of the software itself. A significant amount of time and effort is required to test systems (especially large systems) so it would be cost-effective to implement testing procedures early in the software development rather than postponing it in order to eradicate 'errors' [SEIS]. The general aim of testing is to test a system under the assumption that it contains errors that need to be eradicated. Indeed according to Myers [STGM]:

'Testing is the process of executing a program with the intent of finding errors'

5.2.1 Heuristic Methods

In a heuristic evaluation evaluators (normally experienced programmers) and/or representative users independently examine a product interface and code to determine how well it adheres to a list of heuristics (usability principals). The evaluator goes through the entire product once to get an overview and a feel for the interaction then a

second time to focus on specific parts of the interface and/or code. The following are examples of heuristic evaluation techniques suggested by Jakob Nielsen that could be used [NJUH]:

1. Error prevention

Careful design, which prevents a problem from occurring, is a better solution than having good error messages. This type of testing involves reviewing procedures in the code ensuring that every eventuality has been covered. In Java, the selection statements should include a default to catch unknown events

2. Aesthetic and minimalist design

This suggests that dialogues should not contain information, which is irrelevant or not often required.

3. Help and documentation

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

These heuristic methods provide valuable test results and were therefore implemented in the prototype system.

5.3 Systems Evaluation

It was concluded that certain aspects of the system would require testing to a lesser extent than other aspects of the system. For example the GUI would require testing but not to the degree of testing that would be required for the evaluation of the data content and usability of the technologies implemented. Nevertheless, the testing of the GUI must be completed, as it is the system users' gateway to the functionality of the system. The following tests described in Table 5.1 were carried out in relation to the GUI in accordance with steps outlined by Watts Humphrey¹ [SWMH]:

Test Criteria	Expected Result	Actual Result
Can the windows be resized?	Y	Y
Are the required functions available?	Y	Y
Are the required controls available?	Y	Y
Does each window have a unique title?	Y	Y
Is the use of colour optimised?	Y	Y
Is the use of sound optimised?	Y	Y
Does the window close?	Y	Y

Table 5.1 XML Applications - GUI Evaluation

It is clear that certain areas of functionality must be operational for the GUI to be deemed 'usable' and 'effective'.

5.4 SMIL System Evaluation

5.4.1 GUI Evaluation

Both the interface and functionality of the SMIL presentations were evaluated. The content as well as the interface were considered to be equally important. This is

¹ Software Engineering Institute, Carnegie Mellon University

because SMIL has the potential to help individuals with disabilities (visual, hearing, cognitive/learning) that are hindered from accessing various types of information in various different ways. SMIL also has the potential to work in a CBT environment. The SMIL presentations were tested in the context of the criterion outlined in Table 5.1.

5.4.2 SMIL System Functionality

SMIL was tested with regard to its ability to successfully stream media elements. The synchronisation of the audio with the images was tested as well as the quality of the video (i.e. testing if the sound matched the image). Users were asked to carry out specified tasks within the system and were asked to view the SMIL presentations carefully. They were then asked a series of questions asking them to rate the quality of the presentations with regard to functionality. Figure 5.1 depicts a number of responses to the questions (the maximum points available is 140). Responses indicated that users (in this case 30 individuals) were marginally more satisfied with the video presentation (87.8% average satisfaction rate) than the SMIL presentation (86.4% average satisfaction rate). As the difference is minimal, it is difficult to conclude whether one is more efficient at displaying information than the other. The SMIL synchronisation of audio and video was successful, however the sound quality of the dialogue was marred perhaps due to the audio format incorporated (Audio files, .au). It was necessary to test this aspect of XML in order to evaluate how SMIL could impact on CBTs and Web advancements.

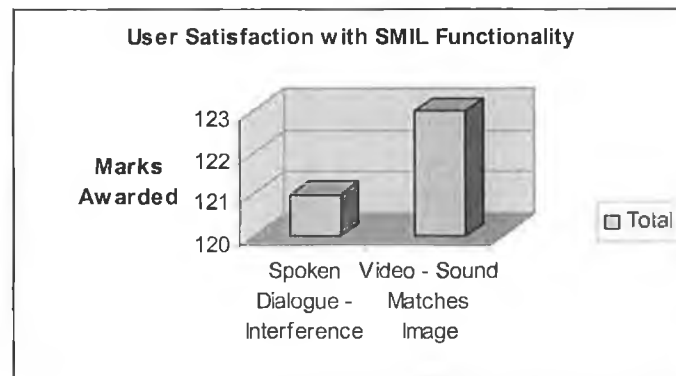


Fig. 5.1 User Satisfaction with SMIL Functionality

The video was shot using a 'Digital Home Camcorder²', was converted into AVI format and generally loaded well during the presentations. A bitrate of 2,300 bps was established to accommodate bandwidth. By testing the presentation with users of varying computer literacy, anomalies were revealed in the presentation. It was discovered that the bandwidth made available was incorrect for the AVI videos incorporated causing one of the presentations to stall on occasions. Therefore, after further investigation, the bandwidth was altered to a value of 1,765 bps (formula: $600/\text{number of minutes/seconds in the video}$).

Any computer connected to a network has a connection bandwidth, which is a maximum speed at which it can receive data. Web users with 28.8 Kbps (Kilobytes per second) modems, for example, can view only those presentations that stream less than 28.8 Kb of data per second. Presentations that stream more data per second may stall because the data cannot traverse the modems fast enough to keep the clips flowing. Successfully targeting the audience's connection bandwidth is vital for producing streamed media. If the presentation falters due to network loading, viewers are less likely to remain interested. The presentations should avoid consuming all of

² Refer to Chapter 4, Table 4.4 for product details

the audience's connection bandwidth. Sufficient bandwidth must be left for network overhead, error correction, re-sending lost data, and so on. Otherwise, the presentation may pause/falter/stall while waiting for more data to arrive.

5.5 VoiceXML System Functionality

In order to test the VoiceXML system efficiently several users of various accents needed to use the system. These individuals (a small sample size of 4 females and 3 males) all had various accents (i.e. Northern Irish, Southern Irish and Russian, all speaking in English). The users relayed their name to the system followed by the sentence "end of recording" or hitting a button on the keyboard to terminate the recording. The users were asked to choose from a menu by selecting a tutorial on "SMIL, SVG or VoiceXML", etc. When the selection was successfully made, the user entered a credit card number using either their voice or the keypad. Testing revealed a time delay anomaly.

If a user were to relay the digits quickly a problem was not encountered, however if the user delayed, then the system would return to the start of the menu again or terminate. This was resolved by increasing the delay allocated to each digit entered. In other words, the user was given more time to relay each digit without the system 'giving up' or 'timing out'. It was revealed that accents did not have a significant impact on the overall functionality of the system. However, as the sample size was relatively small, it is difficult to draw any firm conclusions based on these figures.

5.6 SVG GUI Evaluation

Thirty participants evaluated the interfaces of the SVG pages. It was decided while developing the system that text displays were not as visually pleasing as graphics and animation through SVG. Therefore the pages created were tested in comparison to the same information displayed in XML formatted with a CSS. The data attained from the respondents was noted and is outlined in Figure 5.2.

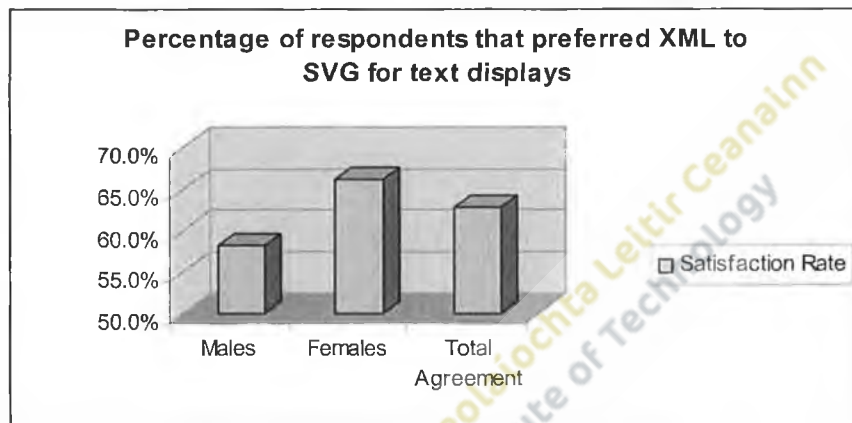


Fig. 5.2 Respondents that preferred XML to SVG for Text Displays

Respondents indicated a preference for the ‘look and feel’ of pages that displayed information through XML (formatted with an external CSS) to the same information displayed in SVG format. 58.3% of the males asked agreed while 66.2% of the females asked agreed, giving a total satisfaction rate with the XML pages of 63%. A suggested reason/explanation for more females preferring the XML page to the SVG page may be a result of varying perceptions.

Males have higher perceptions of ease of reading and font sharpness whereas females focus more on the stylistic differences between the various pages [BMEP]. This suggests that males are less concerned with the style of a Web page than the actual content. Thus, it can be suggested that XML’s ability to ‘style’ a page was more

appealing from a female's point of view. It was clear from the testing of the system that textual displays of information in SVG using the same font size, colour and resolution (96 dpi³) as the XML pages had a significant disadvantage. Users preferred the way in which XML presented the information (i.e. it was generally easier to read).

5.7 The Viability of XML Applications

The viability of combining XML applications to create a 'Web site' was evaluated. This involved the development of an application that combined various markups studied (SMIL, SVG, VML, MathML, XML, XHTML). Each markup had a specific function to carry out either in the form of a tutorial, synchronisation of media elements, the provision of help facilities, animation, etc. (Refer to Chapter 3, Section 3.9.1, Tables 3.3, 3.4 and 3.5). A JDBC-ODBC Bridge was also incorporated into this application that connected to a Microsoft Access database, retrieved information through SQL statements and displayed the results in XML format. The application was subsequently opened in a Web browser.

It was found that depending on the browser used, certain markups were not rendered properly (or in some cases not at all). The connection to the database was successful in both Internet Explorer 5.5 and Netscape 6.2 using a Tomcat Server (Refer to Chapter 4, Section 4.3.4). However the data was displayed in XML format (desired) in Internet Explorer 5.5 but was displayed in ordinary 'text' format in Netscape 6.2.

Further examples include: a plug-in (SVG Viewer) was used in order to display SVG in Internet Explorer 5.5, however the SVG could not be viewed in Netscape 6.2 using

³ Dots Per Inch

the same plug-in. It was also found that an XML document incorporating a Schema and an XSL file could be rendered without error in Internet Explorer 5.5 but not in Netscape 6.2. Again, MathML could be rendered in Internet Explorer 5.5 by using XHTML incorporating an XSL file but this method resulted in the equations not being displayed properly in Netscape 6.2. The problem in displaying the equations in Netscape 6.2 was resolved by using a plug-in (`text/ezmath`) in a HTML file and embedding the markup within it. However this method was not successful in Internet Explorer 5.5. Therefore, it can be suggested that rendering mathematics can be a 'Catch 22' situation.

VML links, graphics and text were displayed without a problem in Internet Explorer 5.5 but only text was displayed in Netscape 6.2. This is mainly as a result of VML being predominantly used with Microsoft based technologies. However, the code used to embed a SMIL file in a HTML file (Refer to Chapter 4, Code Listing 4.1) was successful in both Internet Explorer 5.5 and Netscape 6.2.

The testing revealed that creating Web sites that incorporate several XML applications are possible but not practical at present due to the lack of support that currently exists for these technologies. Plug-ins help but do not work well with several browsers. XML has a place within E-Commerce and CBTs. The applications developed have shown that this is possible but not yet viable due to the lack of support available. These CBTs could incorporate SVG, VML or SMIL (or a combination of all of them). Table 5.2 outlines the various browsers that were incorporated for testing purposes and what was tested on these browsers:

	IE 5.5	Netscape 6.2	Opera 6.0
SVG	Yes	No	No – Requires a plug-in called image/svg+xml
Database Connection using JDBC-ODBC Bridge	Yes	Yes - but data is displayed as text not as XML	No Error: “Could not open file”
SMIL	Yes	Yes	No – Requires a plug-in called audio/x-pn-realaudio-plugin
XML, Schema & XSL	Yes	Not Properly – data is displayed but not formatted	Not Properly – data is displayed but not formatted
MathML	Yes (XHTML & XSL)	Yes - using the text/ezmath plug-in	No – Indicates that it requires the text/ezmath plug-in but the maths are not rendered even with this plug-in in place
Java Applet	Yes	No	No – Error: “The installed version of JRE is incompatible with this version of Opera...”
Data Islands	Yes	No – Data Islands require Java Script for added functionality. Java Script was not recognised.	No – Data Islands require Java Script for added functionality. Java Script was not recognised.
CSS	Yes (For this particular example).	Yes (For this particular example).	Not properly – Did not take into account: background-colour, border:inset, padding or colour (for this particular example).
VML	Yes	Yes - but no links or graphics	Yes - but no graphics and links - gives an error

Table 5.2 XML Applications – Browser Test

5.8 Colour Functionality

In order to have clarity on the pages, it was necessary to carefully consider the colours that were incorporated. Testing revealed that using contrasting colours worked best together.

Colour blindness is something that affects around 7% of males and 0.04% of females [CLRT]. One of the most common forms of colour blindness is the inability to distinguish red from green. In an attempt to satisfy the requirements of any persons using the prototype with a visual disability, the colours within the octagon were taken

into consideration. The statistics from the testing (from a sample size of 30 individuals) revealed a preference for pages that had beige/yellow backgrounds with blue fonts (82% satisfaction rate) as opposed to pages with light grey backgrounds and black fonts (66.6% satisfaction rate).

It can be stated that the clarity of the text improved somewhat with adherence to the octagon. While this aspect has not been extensively researched, it may warrant further research in the future as the choice of colour can affect the clarity of the information displayed. Correct colour usage could also make the information available to a much wider audience.

5.9 Oracle System Timing Issues

The Oracle system developed was tested with regard to the speed at which the processing commands executed. The tests were broken down into two primary areas:

- Time taken by the system to connect to the database.
- Time taken by the system to execute an SQL command after connecting to the database.

Implementing the tests involved putting a timer on the commands using the `java.util.Date()` command. This command obtains the current date and time from the machine, which is running the code. The `getTime()` command can then be used to subtract the `startTime` from the `stopTime` providing a result formatted in milliseconds (ms). Refer to Code Listing 5.1:

```

Java.util.Date startTime=new java.util.Date();
Java.util.Date stopTime=new java.util.Date();
Long elapsedTime=stopTime.getTime()-startTime.getTime();
System.out.println("Elapsed time is " + elapsedTime + "ms");

```

Code Listing 5.1 Putting a timer into Java

The code used to evaluate timing issues was implemented in a number of designated code sections. The results obtained indicated a number of issues, including the impact on the system with regard to an increased data load and the impact of various queries depending on complexity.

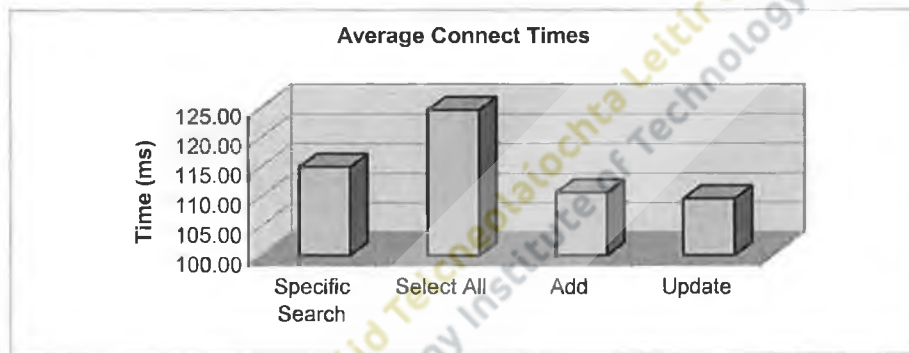


Fig. 5.3 Average Connection Times (Oracle) - 50 XML Files In The Database

In order to maintain accuracy three 'Specific Searches' were carried out, 'Select All' was executed three times, three 'Adds' were executed and three 'Updates' were carried out a total of 15 times (i.e. 45 individual tests). In each instance the 'connection time' was recorded and the average connection time was taken in order to present the information depicted in Figure 5.3. As can be seen, the average connection time ranged from 109.59 ms to 124.59 ms. These average connection times are quick and so could make XML more acceptable when in use with legacy database systems.

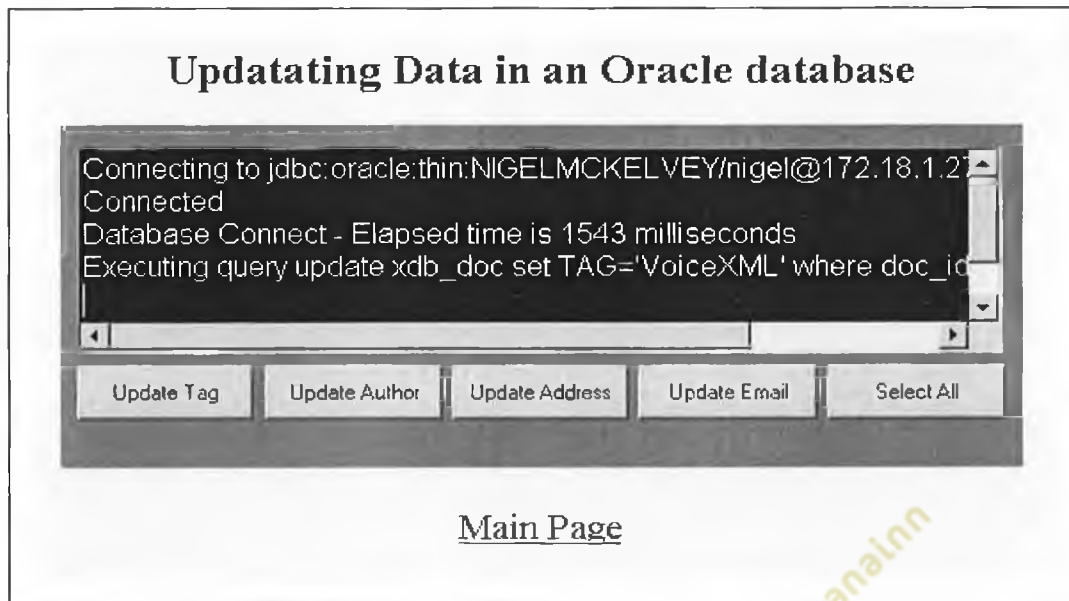


Fig. 5.4 Updating in an Oracle System

It should be noted however that an anomaly was detected during the testing stage. The first query execution after a reboot had a connection time that was significantly higher than subsequent executions. The times ranged from 1,462 ms to 15,263 ms irrespective of the time of day, day of the week, order of execution, etc. Tests were conducted when network traffic was at a high and when there was a reduction in activity. It is suggested that perhaps this is as result of a rebooting problem that exists. However, due to time restrictions a thorough investigation was not conducted.

Therefore, it can be stated that searches do not proportionally increase the connection time. This may be as a result of the parser's ability to filter data not required and search specifically for the data sought by the user.

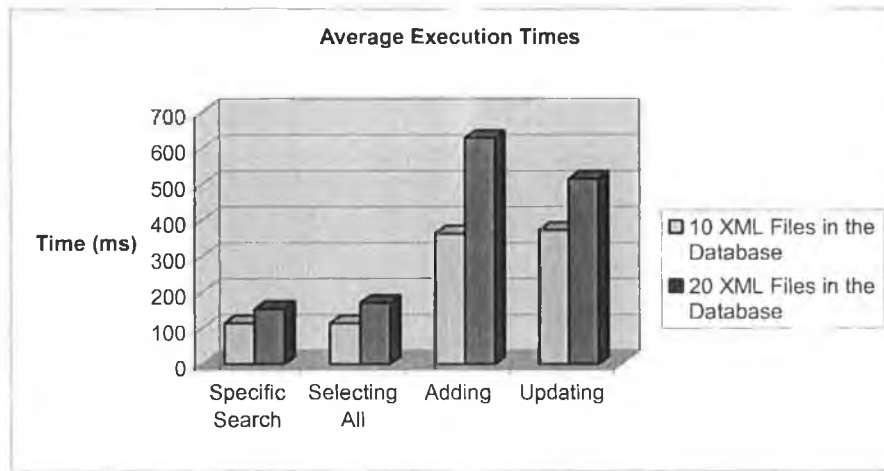


Fig. 5.5 Increased Data Load – Query Execution Times (Oracle System)

Despite the connection times remaining unchanged or improved, the average execution times of the queries increased as can be seen in Figure 5.5. As with calculating the average connection times, an average of three query execution times were recorded 15 times (i.e. 45 individual tests) for each query tested. Query execution times for the ‘Specific Search’, ‘Selecting All’, and ‘Updating’ increased by 26%, 33% and 28% respectively indicating that a ‘Specific Search’ does not increase the query execution time proportionally. ‘Adding’ data to the database with an increased data load resulted in an increase in execution time of 42%.

5.10 MS Access System Functionality

Table 5.3 outlines the various data types that could contain information in a MS Access database. Testing revealed that certain information could not be displayed in XML format. For example, information contained within the data type ‘Memo’ could not be displayed. There are several suggested reasons why the ‘Memo’, ‘OLE Object’ and ‘Hyperlink’ data types were not displayed. One reason could be that these data types contain structured data and this is difficult to access especially when the data

can contain a combination of text or numbers. The size of the data could also affect how it can or cannot be displayed (e.g. 'OLE Object'). It is also worth noting that these data types cannot be indexed. This is a feature that speeds up the search and sorting in a table based on key values. It is also possible that the parser may not be able to evaluate character data in the field. 'Memo' fields that contain HTML or reserved characters may cause the parser to fail for example.

Data Type	Return XML	Return XML Using the WebRowSet class
Text	Y	Y
Number	Y	Y
AutoNumber	Y	Y
Memo	N – Causes error	N – Causes error
Currency	Y	Y
Yes/No	Y	Y
Date/Time	Y	The date 12/09/03 is displayed as: 1063321200000
OLE Object	N – Causes error	Y
Hyperlink	N – Causes error	N – Causes error

Table 5.3 Data Types returned as XML from MS Access via JDBC-ODBC

All of these suggestions would require further investigations in order to resolve the difficulties encountered.

In order to display simple XML it is possible to query a remote JDBC data source and the results are returned as XML. However it is also possible to query the same JDBC data source and return the results as XML using the WebRowSet class. This class provides additional information about the database including connection information required in order to reconnect to the data source for future manipulations. However in the testing of this particular system 'Hyperlink' data types could not be displayed and 'Date/Time' data types could not be displayed properly using the WebRowSet

package. It can be noted nonetheless, that the WebRowSet did allow 'OLE Object' data types to be returned as XML unlike the normal JDBC query.

5.11 Questionnaire Objectives

Questionnaires are commonly used as a part of research to collect data that is used in the research and development of an application and perhaps more importantly to assess the feasibility of the technology itself. Questionnaires allow an application developer to gather information that cannot be found through other means. The purpose of incorporating a questionnaire is to determine users' opinions on the proposed application(s), and then analyse the results. As a result, the developer can then analyse some of the following key questions:

- The performance of the application.
- The users' interaction with the software.
- Users' attitudes towards the software.

The effectiveness of the questions will be determined by the quality and structure of the questionnaire itself. If the questions do not follow a certain standard then the results obtained may be irrelevant, inaccurate or inconclusive as the questions may confuse respondents.

When evaluating the software developed it was necessary to investigate and determine the objectives of the questionnaire in order to focus on the aspects of the system that required a level of evaluation and/or testing. The following objectives were deemed appropriate:

- To determine if the user(s) could interact with the application(s) successfully.

- To determine if XML could be successfully incorporated within a system (i.e. could all the various XML technologies be combined into one seamless package).
- To determine if there was an adequate help facility.
- To determine if the application(s) were aesthetically pleasing.
- To determine if the application(s) were technically capable of executing a set of assigned tasks.
- To determine how quickly a user could carry out a set of tasks.
- To determine if the user could understand the tutorials.
- To obtain sufficient information in order to successfully evaluate the software and assimilate the results.

5.12 Questionnaire Design Guidelines

It was necessary to follow certain guidelines when designing the questionnaire in order to: keep the questionnaire as relevant to the objectives outlined initially as possible and also to ensure that the questionnaire be similar to those used in industry. In order to maintain the standard of the questionnaire, meetings were held with an experienced lecturer in the Business/Marketing department of the college. The respondents required to answer the questionnaire and perform the tasks should not feel that a specific answer is required (i.e. the questions should not be leading). Therefore, it was very important that the questionnaire be structured in a manner that would conform to some basic design guidelines. The following are a number of the guidelines that were adhered to when developing the questionnaire:

- Structure questionnaire logically; Have a beginning, middle and end.
- Questions should go from the general to the specific.

- Consider the age of respondents.
- Consider the respondents' attention span.
- Keep questions short; Less than twenty-five words.
- Do not lead with a response. For example asking the question, "*Do you often...*"
- Avoid jargon.
- Avoid double-barrelled questions. For example, "*Do you like the taste and aroma of tea?*"
- Do not antagonise respondents.
- Evaluate if the respondent would be able to answer the question.
- Give the respondents time to answer questions.
- Consider using open-ended questions.
- Consider the aesthetics of the questionnaire.
- Distinguish questions from answers.
- Consider using questions asked in both the positive and negative.
- Use check boxes as opposed to blank lines.

5.12.1 Likert Scale

Whilst keeping the guidelines discussed in Section 5.12 in mind, the Likert Scale was considered the most appropriate technique to employ when designing the questionnaire.

The Likert Scale is a measurement scale with five response categories ranging from 'strongly agree' to 'strongly disagree' [MTRM]. The respondents are required to

indicate a degree of agreement or disagreement with each of a series of statements about the application(s). For example:

Do you agree that the software was easy to use?

Strongly	Somewhat	Neither	Somewhat	Strongly
Agree	Agree	Dis/Agree	Disagree	Disagree
[]	[]	[]	[]	[]

As stated in Section 5.12 it was necessary to create a questionnaire that would not lead the respondents. The Likert Scale helps bring a sense of balance to the questionnaire as it has an equal number of favourable and unfavourable categories, which leaves the questionnaire generally more objective [MTRM]. The Likert Scale also allows a respondent to have a neutral response, i.e. 'Neither Dis/Agree'.

5.12.1.1 Advantages of the Likert Scale

- It is easy to construct and administer [MTRM].
- Respondents generally understand how to use the scale.

5.12.1.2 Disadvantages of the Likert Scale

- It takes longer to complete because respondents have to read and understand each question.

5.13 Conclusions

The prototype systems developed were tested for completeness, correctness, reliability and acceptance. Attention was also paid to the applications' GUI as colour and Web accessibility for individuals with various disabilities were taken into consideration. These tests were carried out at various stages in the development of the system. Tests included the testing of the functionality of the XML applications developed and their interactions with each other. XML's interactions with legacy databases were also tested with regard to data loading and query complexity. The timing issues that arose from this testing provided data on XML's capabilities within relational databases and its potential impact on legacy systems. The information gathered enabled a hypothesis to be proposed which is further documented in Chapters 6 and 7.

The comparison of an SVG file with a VML file in Chapter 2, Section 2.12.3 revealed that for that particular example, the SVG file was smaller in size. This suggests that XML is striving to reduce file sizes in an attempt to bring multimedia elements to the Web more quickly. Research conducted revealed that when a Bitmap graphic (49.2KB in size) was embedded in a MS Word document and saved as a Web page (i.e. with the .html extension), a VML file was created. The Bitmap graphic was converted to a PNG graphic 33.8KB in size (a 31.3% reduction in file size). Along with the new PNG graphic, an XML file and the VML file were created. The VML file was 3.07KB in size while the XML file was 226 bytes in size. The original Word document itself was 95KB in size, which suggests that even when the PNG file, the XML file and the VML file are added together, their total file size is still considerably less than the original Word document.

However, further research considered the situation where a GIF graphic (810 bytes in size) was embedded in a Word document and saved resulting in a document 23KB in size. When the document was saved as a web page the graphic remained a GIF but increased in size to 840 bytes in size, which was a 3.7% increase. However, taking into consideration the VML file created (2.83KB in size) and the XML file (164 bytes in size), the overall size is reduced. It is also worth noting that VML is predominately used with Microsoft, which implies that incorporating VML into a system that does not compatible with Microsoft would be unsuccessful.

The next chapter includes conclusions based upon the research conducted.

lyit | Institiúid Teicneolaíochta Leith Ceanaínn
Letterkenny Institute of Technology

Chapter 6

Summary & Conclusions

lyit | Institiúid Teicneolaíochta Leitir Ceanaínn
Letterkenny Institute of Technology

6.1 Preface

This chapter allows us to draw conclusions about the capabilities of XML and its interaction with legacy databases with regard to the applications developed. Section 1.1.2 in Chapter 1 defines two main problems that required further investigation. From the research conducted and the applications developed, these ‘problems’ were encountered and conclusions have been drawn as a result. A summary of the research is also outlined in this chapter.

6.2 Summary

This thesis provides an outline of the evolution of XML technologies and also provides a broad description of six of these technologies: SVG, VML, MathML, SMIL, VoiceXML and SpeechML (used as a theoretical comparison with VoiceXML – refer to Chapter 2, Sections 2.10 to 2.16). Each of these technologies were then described in terms of their purpose, capabilities, advantages and disadvantages. Research indicated that there has been a limited amount of testing with regard to XML and related technologies, working alongside each other simultaneously. Therefore, a review of relevant literature was conducted which outlined the theory behind XML and the chosen applications of XML. In addition to the six technologies researched, numerous others exist but were not investigated in this thesis due to time constraints.

An application was created that combines all these technologies (except VoiceXML – required a separate application, refer to Chapter 4, Section 4.2.2). It was created with the intention of exploring the capabilities of XML and also to demonstrate how these technologies could be used in a CBT environment. By combining these technologies

the capabilities of the technologies could be analysed with regard to portability, level of available support, multimedia networking, bandwidth improvements and Web site advancements, etc. The system was designed in such a way as to steer the applications toward providing the best possible means of communicating information. It was also created with the intention of improving efficiency and revealing the merits/demerits of utilising these technologies considering the level of support (or lack thereof) currently available.

In addition, it was necessary to evaluate how XML interacts with legacy database systems. Two databases were chosen in order to demonstrate how large corporate organisations could utilise XML as well as smaller to medium enterprises. As a result, Oracle (for larger organisations) and MS Access (for smaller businesses) were chosen. They incorporated the use of a Thin Driver and the JDBC-ODBC Bridge respectively. A feasibility study (refer to Chapter 5, Sections 5.9, 5.10 and 5.13) was then conducted in order to determine if incorporating XML with legacy databases would be advantageous and what implications (if any) existed in order to do this. Prior to this implementation, the systems were designed in such a way that accurate results could be obtained from the testing and evaluation of the prototype. The testing of these systems raised issues regarding the impact of XML on database performance as well as issues regarding the Middleware incorporated.

The capabilities of VoiceXML were demonstrated through a prototype application ("Kappa-C") allowing potential users to interact with the system through a series of voice driven commands. The purpose of this application was to investigate the capabilities of VoiceXML with regard to CBTs and basic functionality. Users were

required to enter credit card details, which were verified by the system, by using Java Script. When correct credit card details were entered, the user proceeded to an array of tutorials, which were accessible through a voice driven menu. The testing phase highlighted a number of issues from grammar pronunciations to timing issues as well as the impact of accents on systems. Also noted were compatibility issues with the other XML technologies incorporated (Refer to Sections 6.3.3, 6.3.6, 6.3.6.1, 6.3.7.1 and Chapter 7, Section 7.1.3).

Testing of the systems developed provided indications as to the capabilities of XML and its interaction with legacy databases. As a result a series of conclusions are summarised with recommendations stipulated as to further work that could be conducted in order to further investigate XML.

6.3 Conclusions

The following section outlines the conclusions that have been drawn based on the research conducted:

6.3.1 XML is a Database in itself

From the research conducted (i.e. from books, white papers, technical reports, journals, web articles, etc.), XML documents were often referred to as ‘databases’ themselves. In other words, data can be structured in such a way that it can be easily accessed by other means. This was verified as information stored in an XML document can be retrieved using both a Java Applet and Data Islands. However, for numerical data to be manipulated for example, Java or Java Scripts need to be

incorporated for additional functionality. Not all of the browsers that were used for testing purposes supported this 'functionality' (Refer to Chapter 5, Section 5.7).

6.3.2 Oracle is difficult to implement

Difficulties were encountered when manipulating the Oracle database. These difficulties were mainly due to the use of Oracle itself. Oracle has been seen to be a rather cumbersome system to implement. In order to successfully load XML documents into the database, Java was required to take the XML document, remove the data contained within the tags and put the data into the relevant tables in Oracle. It was necessary to log into Unix as the Oracle Administrator. A further requirement was to load the TNS listener, check its status in order to retrieve the services it supports and start it (Refer to Chapter 3, Section 3.6.3.2).

6.3.2.1 Java and XML can successfully interact with a database

Research revealed that XML can be stored in a relational database (Oracle8i) and placed in tables for manipulation. It was also shown that XML could be retrieved from a relational database (MS Access 2000) and displayed as such. Therefore, it was proven, that when combined with Java, XML could successfully interact with a database.

6.3.3 JDBC-ODBC Middleware Limitations

XML cannot directly access data from legacy databases, which is a significant drawback of the technology. Access requires the use of middleware thus it is susceptible to all the limitations of the middleware applied. In the case of this thesis the middleware used was the JDBC-ODBC Bridge. Some of the limitations revealed

included certain data types not being properly serialised as XML (i.e. returned as XML) (Refer to Chapter 5, Table 5.3). However, there is more management information available through returning the data as XML. For example, returning the data as XML includes connection information required in order to reconnect to the data source for future manipulations, thus speeding up reconnection times (Refer to Chapter 5, Section 5.10).

6.3.4 XML requires significant formatting

XML in general involves a significant amount of formatting through the use of XSL, CSS, XSLT, DTDs, Schemas, etc. (Refer to [NMKa] for further analysis). In addition, if a web site developed in XML enables users to make their own tags, this would require XML Namespaces to be implemented in order to manage the extensibility that XML possesses. The format of buttons and scrollbars however may be customised through the use of CSS. Again, the viability of creating such Web sites came into question when the lack of support for these 'formatting capabilities' arose during the testing phase (Refer to Chapter 5, Section 5.7).

6.3.4.1 DTDs and XML Schemas provide a similar function

It was noted that DTDs and XML Schemas provide similar functionality. Even though XML Schemas are newer than DTDs, they do not consider the 'ENTITY', which is an important aspect of most XML documents. This can be considered a drawback as entities are often used for breaking down a larger document into smaller units. Therefore, if certain entities are not specified then the document cannot be broken down and as a result the filtering of data can be hindered (Refer to Chapter 2, Sections 2.3.1 and 2.7.3). It is this 'filtering' effect that is advantageous to E-Commerce

applications as only the required information is returned.

6.3.5 File Sizes

With any Web development strategy, striving to reduce file sizes is always an important requirement. Research conducted showed how XML applications are consequently helping to reduce file sizes. Practical implementations of these applications revealed some figures that supported this hypothesis as well as revealing some previously undocumented aspects such as those discussed in Sections 6.3.5.1, 6.3.5.2 and 6.3.8 (Refer also to Chapter 2 Sections 2.11, 2.12, 2.12.3, 2.14, 2.16).

6.3.5.1 SVG versus VML

Applications were written in both SVG and VML to display a geometric shape. The same attributes were used to describe this shape in both instances. The results revealed that the SVG file was approximately 14% smaller in file size in comparison to the VML file. This was further verified with a second example, which took two different geometric shapes and applied the same attributes to each. The results revealed that in the second instance the SVG file was 39% smaller in size. SVG offered a further advantage in that VML is essentially a Microsoft product. In other words, unless Internet Explorer is being used, VML will not display properly. Testing revealed that graphics and links were not properly rendered in a non-Microsoft browser (Refer to Chapter 5, Section 5.7).

6.3.5.2 SVG improves bandwidth

Generally graphics created in SVG result in a smaller file size in comparison to creating a HTML file that refers to GIFs or JPEGs, etc. Therefore, SVG could help

alleviate the problem of slow download times. However, the amount of code required in order to depict an image in SVG appears to be excessive. For example, creating a gradient effect for a background could take up to 25 lines of code (depending on the desired result).

6.3.5.3 Text displayed using SVG appears ‘smudged’

SVG proves a good tool for manipulating graphics, however, it proves inefficient when displaying text. Text displayed through SVG seems ‘smudged’. The clarity of the display varied with the choice of colour and resolution incorporated. VML also proves a good tool for manipulating graphics, however, text displayed through VML is significantly clearer than that displayed through SVG. Material printed from a VML page also appeared to be clearer than that from an SVG page. However, the display of text through formatted XML is significantly clearer than through VML or SVG. Figure 6.1 depicts the word ‘Country’ rendered using SVG, XML and VML. The same formatting specifications were applied (i.e. a beige background, blue text, a font type of Arial Sans-Serif, a font size of 14 and a normal font weight).

SVG	XML	VML
Country	Country	Country

Fig. 6.1 Visual Differences When Rendering Text

It can also be seen from Figure 6.1 that the three words look very different visually despite the fact that they are all of size 14 font. A significant difference was also noted in the print quality from each of the technologies outlined in Figure 6.1. XML and VML pages printed much more clearly than the SVG pages.

6.3.5.4 SVG as an alternative to current technologies

The use of SVG with timers could be considered a viable alternative to PowerPoint. However when SVG and timers were combined, neither animation nor links were supported. The provision of navigation capabilities (first slide, last slide, etc.) would require the use of Java Script or similar code, which makes systems development more difficult as opposed to easier.

6.3.5.5 SVG and VML can provide further alternatives

SVG may prove a viable alternative to Flash as specific software is not required and timers can be incorporated into the code. Microsoft PowerPoint also possesses the ability to save presentations as HTML, including the automatic conversion of pictures to VML thereby helping to reduce file sizes (Refer to Chapter 2, Section 2.11).

6.3.6 Incorporating SMIL requires media considerations

SMIL is a user-friendly protocol providing the correct browser and media elements are installed. From Table 2.5 in Chapter 2, it can be seen that movie files (.MOV) are available through RealPlayer G2, however research indicates a lack of support for this file type. When the presentation was run incorporating a .MOV file an error resulted, "RealOne Player needs to download new software to play this clip". The clip was eventually successfully presented by saving the same SMIL file with the .mov file extension. This caused the browser to use the QuickTime plug-in to handle the file. However, the eight-character string *SMILtext*, needed to be added to the beginning of the file so QuickTime knew what it was. CBTs would be augmented through the use of SMIL technology in the synchronisation of audio and visual elements.

6.3.6.1 SMIL can be portable

SMIL is portable but only if bandwidth has been considered carefully. One presentation could require up to five different file types, thus increasing complexity. An efficiently streamed SMIL presentation requires the files' relationships to be arranged and maintained, even whilst publishing. When changes are made to a presentation, those changes must often be reflected in multiple files. This places limitations on the portability of SMIL presentations (Refer to Chapter 5, Section 5.4.2).

6.3.6.2 Applying the correct bitrate in SMIL is important

Successfully targeting the audience's connection bandwidth is vital for producing streaming media. If the presentation falters because of inadequate bandwidth availability, then viewers will lose interest. The presentations must avoid consuming all of the audience's connection bandwidth. Sufficient bandwidth must be left for network overhead, error correction, re-sending lost data, etc. (Refer to Chapter 5, Section 5.4.2 and [NMKd] for further analysis).

6.3.7 VoiceXML is a viable option for CBTs

VoiceXML requires IBM's WebSphere Voice Toolkit and IBM WebSphere SDK Web server. Both of these technologies are comparatively difficult to install and administer. Memory requirements are also significant and additional hardware is required (Refer to Chapter 4, Table 4.3). VoiceXML could enhance CBTs, however, it does not prove viable for remote administration. One of the major advantages of the use of VoiceXML is that it helps widen the availability of data to visually impaired system users. Refer to [NMKc] for further analysis of the topic.

6.3.7.1 SALT will enhance Web applications

A multi-modal system is one where a user can interact with Web and voice content at the same time. SALT tags can help bring this about. However SALT is still relatively new and as of yet support is quite limited (refer to Chapter 2, Sections 2.16 and 2.16.3).

6.3.8 MathML reduces file sizes

Previously it was necessary for mathematicians to display notations by embedding screenshots of notations, created in various applications, such as GIFs or JPEGs. Inevitably the file sizes of these documents were increased and the download time on Web pages were also increased.

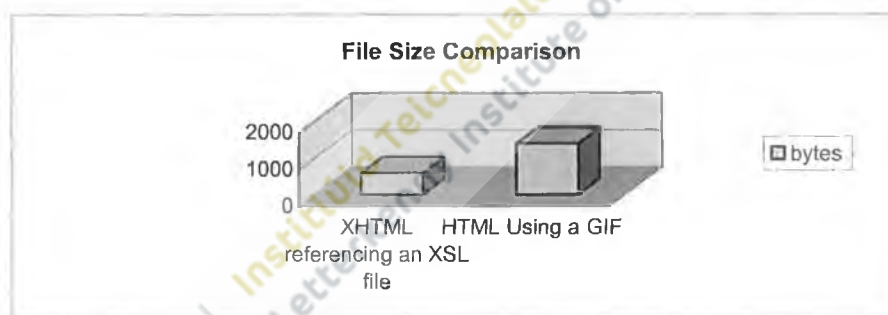


Fig. 6.2 File Size Comparisons For Rendering Maths

If a mathematician wanted to render an equation on the Internet it would have been necessary to create a snapshot of the equation in either GIF or JPEG format. A GIF was created (1,160 bytes in size). This was then embedded in a HTML page (232 bytes in size). This gave a total of 1,392 bytes of information. Alternatively, an XHTML file could be created (591 bytes in size) which could reference an XSL file from a URL and display the same notation in the form of markup. The difference in file size for transport across a network is substantial as can be seen in Figure 6.2. Refer to Chapter 2, Section 2.14 and reference [NMKa] for further analysis.

Chapter 7

Recommendations for Further Research

lyit | Institiúid Teicneolaíochta Leitrí Ceanáin
Letterkenny Institute of Technology

7.1 Recommendations

From the research carried out, a number of issues arose which merit further research. However due to time restrictions these issues were not further researched. In addition, a number of enhancements to the prototype system(s) are suggested.

7.1.1 Further development of the JDBC-ODBC Prototype

During the testing and evaluation phase of the prototype software, it became apparent that certain data types (e.g. OLE Object, Memo and Hyperlink) caused an error when the data was serialised as XML (Refer to Chapter 5, Table 5.3). It was determined that if the system could serialise these data types, it would have enhanced functionality. The development of an enabling technology could significantly enhance the capabilities of XML.

7.1.2 Further development of the Oracle Prototype

In order to improve the user interface of the Oracle prototype the user should be able to specify through a text field the URL to an XML file, which they wish to upload into the database. This would eliminate the need to use a batch file. Time restrictions meant that this particular feature was not further developed.

7.1.3 Further development of the VoiceXML Prototype

Although the VoiceXML prototype revealed the potential of a voice driven Internet, the application was a separate one and so could not be tested for compatibility with other XML technologies. Therefore, additional research into SALT tags (Specification released in July 2002) should be undertaken in an attempt to create a

multi-modal application that could be embedded in a Web page along with SVG or SMIL for example. It was also noted that literature was very difficult to find on the topic. Many sources merely discussed the potential that VoiceXML offers. Thus, it is felt that further research should be carried out in this area.

7.1.4 Research into the advancement of GIS with SVG

The advantages offered by SVG for improved Internet efficiency has led to the advancement of other technologies. GIS systems for example, are employing the capabilities of SVG for improved pan/zoom capabilities and reduced file sizes (although, not for all files, see Section 7.1.5). SVG is also making these GIS systems available to a wider audience via in-car displays and PDAs. This is an area, which warrants further investigation in order to further evaluate the capabilities offered by SVG.

7.1.5 Research into the capabilities of SVG for rendering photograph quality images

At present it does not appear that SVG will replace the need for 'photograph quality' images. Perhaps, this limitation should be further researched due to the increasing requirement of reducing the file sizes of images before placing them on a Web site.

7.1.6 Research into XML's capabilities for an improved Internet

Improving the speed at which data is transmitted across the Internet is proving to be an important aspect to data communications. Research conducted in the field has shown how XML has the ability to help reduce file sizes and format data for display purposes. The presentation of a paper ([NMKa]) generated considerable interest

(questions from the Conference Chairperson and conference attendees) in the area thus suggesting, that the interest shown in relation to this aspect of XML warrants further investigation.

7.1.7 Research new/other XML technologies

Time restrictions meant that not all of XML's technologies were researched in this thesis. Technologies such as Chemical Markup Language (CML), Document Style Semantics and Specification Language (DSSSL), Nuance Grammar Specification Language (GSL), Java Speech Markup Language (JSML), Petri Net Markup Language (PNML), Wireless Markup Language (WML), etc. were not researched. Whilst these technologies were not directly relevant to the prototypes developed for this particular thesis, they do provide a number of additional XML capabilities that warrant investigation.

Appendix A

References

Note

<u>ACM:</u>	<u><i>The Association for Computing Machinery</i></u>
<u>IEEE:</u>	<u><i>The Institute of Electrical and Electronic Engineers</i></u>
<u>IEI:</u>	<u><i>The Institution of Engineers of Ireland</i></u>

- ACMK Kiyomitsu, Hidenari, et al. “ActiveWeb: XML-Based Active Rules for Web View Derivations and Access Control”. *ACM International Conference Proceeding Series - Proceedings of the workshop on Information technology for virtual enterprises*. Publisher: IEEE Computer Society Press. USA. January 2001. Volume 23, Issue 6. ISBN: 0-7695-0960-6. Pp. 31 – 39.
- ACML Lucas, Bruce. “VoiceXML for Web-Based Distributed Conversational Applications”. *Communications of the ACM*. Volume 43, Issue 9. ACM Press, New York. September 2000. ISSN – 0001-0782. Pages 53-57.
- ACMM Marriott, Kim, et al. “Fast and Efficient Client-Side Adaptivity for SVG”. *International World Wide Web Conference. Proceedings of the Eleventh International Conference on the World Wide Web*. ACM Press, New York. May 2002. ISBN – 1-58113-449-5. Pages 496-504.
- ACMR Routledge, Nicholas, et al. “UML and XML Schema”. *Australian Computer Science Communications, Proceedings of the Thirteenth Australian Conference on Database Technologies – Volume 5*. Australian Computer Society, Inc. Volume 24, Issue 2. ISBN ~ ISSN: 1445-1336, 0-909925-83-6. Pages 157-166.
- ACMS Laird, Cameron. “XSLT Powers a New Wave of Web Applications”. *Linux Journal*. Volume 2002, Issue 95. Specialised Systems Consultants, Inc. USA. ISSN 1075-3583.
- ACMX Grundy, John, et al. “Building Multi-Device, Component-Based, Thin-Client Groupware: Issues and Experiences”. *ACM International Conference Proceeding Series - Third Australasian Conference on User Interfaces - Volume 7*. Publisher: Australian Computer Society, Inc.. Australia. January 2002. Volume 24, Issue 4. ISBN~ISSN: 1445-1336, 0-909925-85-2. Pp. 71 – 80
- AXML Ceponkus, A. & Hoodbhoy, F. “Applied XML - A Toolkit for Programmers”. WILEY. USA. Book & CD Edition. July 1, 1999. ISBN: 0-4713-44028.
- BATK “Batik Overview”. The Apache Software Foundation. Contact: Roderic Olvera Young at rododyboston@medione.net . Available Online at: <http://xml.apache.org/batik/#BatikApplications>.

- BCMI Shim, Simon S.Y. et al. "Business-to-Business E-Commerce Frameworks". *IEEE Computer*. Vol. 33, No. 10. October 2000. pp. 40-47.
- BMEP Bernard, Michael L., et al. "Examining Perceptions of online text size and typeface legibility for older males and females". *Proceedings of the sixth Annual International Conference on Industrial Engineering. Theory, Applications, and Practice*. USA. 2001.
- BTBF Fitzgerald, Michael. "Building B2B Applications with XML - A Resource Guide". WILEY. Canada. 2001. ISBN: 0-471-40401-2.
- BWXL St. Laurent, Simon. "Browser XML Display Support Chart". 2nd May 2000. Available online at: <http://www.xml.com/pub/a/2000/05/03/browserchart/index.html>,
- BXAL St. Laurent, Simon & Cerami, Ethan. "Building XML Applications". USA. 1999. McGraw-Hill. ISBN: 0-07-134116-1.
- BXML Hunter, David et al. "Beginning XML". 2nd Edition. WROX. USA. 2000. ISBN: 1-861003-4-12.
- CARH Hoare, Charles, A. R. (*Professor of Computation at the University of Oxford, England*). "The 1980 ACM Turing Award Lecture". *Communications of the ACM*. Tennessee. February 1981. Volume 24, Number 2. Available Online at: <http://www.braithwaite-lee.com/opinions/p75-hoare.pdf>
- CCMI Grado-Caffaro, Maria-Angeles & Grado-Caffaro, Martin. "The Challenges that XML Faces". *IEEE, Computer*. Vol. 34, No. 10. October 2001. pp. 15-18.
- CLRT Colour Therapy. Available Online at: http://www.colourtherapyhealing.com/colour/complementary_colours.asp. Contact: infor@colourtherapyhealing.com
- CMPS Selic, Bran. "A Generic Framework for Modelling Resources with UML". *IEEE, Computer*. Vol. 33, No. 6. June 2000. Pp. 64-69.
- CMPY Junichi Suzuki & Yoshikazu Yamamoto. "Leveraging Distributed Software Development". *IEEE, Computer*. Vol. 32, No. 9. September 1999. Pp. 59-65.
- CSLT Duffy Marsan, Carolyn. "Shakin' on the SALT". *Network World*. 23rd September 2002. Available Online at: <http://www.nwfusion.com/buzz/2002/salt.html>
- DBMSB Butler Group. "Database Management Systems – Managing Relational and XML Data Structures". *Technology Evaluation and*

- Comparison Report*. September 2002. Volume 3. ISBN 0-9542845-2-6. UK.
- DDJH Herzberg, Amir. "Securing XML". *Dr. Dobb's Journal. Scientific & Engineering Computing*. Volume 27, Issue 3. March 2002. pp. 56.
- DDJK Kanalakakis, John M., Jr. "Web Services and Java Server Pages". *Dr. Dobb's Journal. Web Services*. Volume 27, Issue 1. January 2002. pp. 28.
- DDJN Naccarato, Giuseppe. "XSLT Querying and XML Documents". *Dr. Dobb's Journal. Database Development*. Volume 27, Issue 12. December 2002. pp. 24.
- DDJY Yudkowsky, Moshe. "Voice Biometrics and Application Security". *Dr. Dobb's Journal. Computer Security*. Volume 27, Issue 11. November 2002. pp. 16-22.
- DEAB Berg, Clifford J. "Advanced Java 2. Development for Enterprise Applications". 2nd Edition. Sun Microsystems Press. Prentice Hall. ISBN 0-13-084875-1. USA. 2000.
- DOML Lerner, Reuven M. "At the Forge – XMLC". *Linux Journal*. Volume 2001, Issue 8. August 2001. Specialised Systems Consultants Inc. USA. ISSN: 1075-3583.
- DSBM McMullin, Barry. "Users with Disability Need Not Apply? Web Accessibility in Ireland". *First Monday*. Volume 7 Number 12. December 2002. Available online at: <http://eaccess.rince.ie>
- DTDG Johnson, M. "The Golden DTD Rules – XML in practice". 15th Feb 2001. Available Online at: http://www.itworld.com/nl/xml_prac/02152001/
- DTDX Johnson, M. "Make Up a Markup – XML in practice". 25 Jan 2001. Available Online at: <http://www.itworld.com>
- DXDB Graves, Mark. "Designing XML Databases". Prentice Hall Inc. 2002. USA. ISBN: 0-13-088901-6.
- ECOM "The Business Forum. Electronic Commerce Explained". Microsoft Corporation. *White Paper*. Available Online at: <http://www.bizforum.org/whitepapers/microsoft.htm>
- ERDB Budgen, David. "Software Design". Addison-Wesley. 1994. UK. Consulting Editor: A.D. McGettrick. ISBN: 0-201-54403-2. Pp. 104-108.

- FBTB Metcalfe, D. et al. "The Future of Europe's Online B2B Trade". July 2002. Available Online at: <http://www.forrester.com/ER/Research/Report/Summary/0,1338,15378,00.html>
- GMLC Goldfarb, Charles F. "The Roots of SGML – A Personal Recollection". 1996. Available Online at: <http://www.sgmlsource.com/history/roots.htm>
- GSYL Lexico Publishing Group, LLC. <http://dictionary.reference.com/search>.
- HPWD Taylor, Humphrey. "How The Internet Is Improving The Lives Of Americans With Disabilities". *The Harris Poll*. June 2000. Available Online at: http://www.harrisinteractive.com/harris_poll/index.asp?PID=93
- HTMLB Bosak, Jon. "XML, Java, and the future of the Web". Sun Microsystems. *White Paper*. 2nd October 1997. Available Online at: <http://www.xml.com/lpt/a/w3j/s3.bosak.html>.
- HTMLF Freter, Todd. "XML: It's the Future of HTML". Sun Microsystems. April 2000. *White Paper*. Available Online at: <http://www.sun.com/980602/xml/>
- ITPC Coyle, Frank P. "Breathing Life Into Legacy". *IEEE, IT Professional*. September/October 2001. pp. 17-22.
- JAVB "Building Business-to-Business Applications - Today's Big Web Opportunity". Sun Microsystems, Inc. Available Online at: <http://java.sun.com/xml/b2b.html>
- JDBR Reese, George. "Database Programming with JDBC and Java". 1st Edition. O'Reilly. June 1997. Editor: Andy Oram. USA. ISBN: 1-56592-270-0.
- JVCB Darwin, Ian F. "Java Cookbook". O'Reilly. First Edition. ISBN: 0-596-00170-3. June 2001.
- KNOG Klever, N. "Online Graphics with SVG". 2002. Available Online at: <http://www3.cti.ac.at/icl/archive/presentation/klever.pdf>
- LNVS Leavitt, Neal. "Two Technologies Vie for Recognition in Speech Market". *IEEE, Computer*. June 2003. Vol.36, No. 6. pp. 13-16.
- LSGX Gould, Lawrence S. "XML: The Latest iTechnology – Internet and Integration". Available Online at: <http://www.autofieldguide.com/articles/030004.html>

- MCRI Maxwell, Christine. "Global Trends that will Impact Universal Access to Information Resources". UNESCO. July 15th, 2000. Available Online at: <http://www.isoc.org/isoc/unesco-paper.shtml>
- MDMM Marshall, Dave. 2001. Available Online at: <http://www.cs.cf.ac.uk/Dave/Multimedia/node99.html>
- MMLB Kamthan, P. "The State of MathML: Mathematically Speaking (and Stuttering)". March 2000. Available Online at: <http://www.tech.irt.org/articles/js208/index.htm>
- MMLC Cool, Thomas. "The Disappointment and Embarrassment of MathML - update: Including Reactions and Answers". April 11th, update April 16th 2000. Report 2000-04-11 & 16. JEL A00. Available Online at: <http://www.dataweb.nl/~cool/Papers/MathML/OnMathML.html>.
- MMLD Topping, Paul. "Mathematics on the Web: MathML and WebType". *White Paper*. Design Science, Inc. 21st January 1999. Available Online at: http://www.dessci.com/en/reference/white_papers/mt_mathml.htm
- MMLO XSLT stylesheets for MathML. "Current Browser Support". Available online at: <http://www.w3.org/Math/XSL/Overview-tech.html>
- MMLW World Wide Web Consortium. 7th July 1999. Available Online at: <http://www.w3.org/TR/REC-MathML/>,
- MMUS Mudawwar, Muhammad F. "Multicode: A Truly Multilingual Approach to Text Encoding". *IEEE, Computer*. April 1997. Volume 30, Number 4. pp. 37-43.
- MRKM Morento, Pedro J, et al. "From Multimedia Retrieval to Knowledge Management". *IEEE, Computer*. Vol. 35, No. 4. April 2002. pp. 58-66.
- MTRM Malhotra, Naresh K. "Market Research, An Applied Orientation". Third Edition. Prentice Hall. 1999. USA. ISBN: 0-13-083044-5.
- NCCB Newman, Chuck. "Considering the Color-Blind". 2000. Available Online at: <http://www.webtechniques.com/archives/2000/08/newman/>
- NJUH Nielsen, Jakob. "Ten Usability Heuristics". Available Online at: http://www.useit.com/papers/heuristic/heuristic_list.html
- NMKa McKelvey, Nigel et al. "Formatting – For Improved Internet Performance". Proceedings of the *IEI/IEEE Irish Telecommunications Systems Research Symposium*. May 2003.

- NMKb McKelvey, Nigel et al. "XML Applications and how they interact in a multimedia networking environment". *Proceedings of the First Joint IEI/IEEE Symposium on Telecommunication Systems Research*. November 2001.
- NMKc McKelvey, Nigel et al. "Representing Human-Computer Dialogues Using VoiceXML". *Proceedings of The Irish Signals and Systems Conference 2003*. July 2003.
- NMKd McKelvey, Nigel et al. "SMIL - Portability and Browser Support Issues". *Proceedings of The 32nd International Conference on Computers and Industrial Engineering*. August 2003.
- NMKf McKelvey, Nigel et al. "XML - More Than Just A Buzzword". *The Irish Scientist Year Book 2002*. November 2002.
- NMKg McKelvey Nigel et al. "XML - The Rejuvenator!". *The Irish Scientist Year Book 2003*. November 2003.
- NMSX Naedele, Martin. "Standards for XML and Web Services Security". *IEEE, Computer*. April 2003. Volume 36, Number 4. pp. 96-98.
- NWMG Myung-Sup Kim, Mi-Joung Choi, James W. Hong. "A Load Cluster Management System Using SNMP and Web". *International Journal of Network Management*. Volume 12, Number 6. November/December 2002. Publisher: John Wiley & Sons, Ltd. USA. ISSN: 1099-1190.
- ORLG Gennick, Jonathan et al. "Oracle8i DBA Bible". IDG Books Worldwide. 2000. USA. ISBN: 00-7645-4623-6.
- PDBW Williams, Kevin et al. "Professional XML Databases". Wrox Press Ltd. 2000. Published in UK. Printed in Canada. ISBN: 1-861003-58-7.
- PTCI Campbell, Mary K. "Standard Politics". *IEEE Potentials*. October/November 2002. Volume 21, Number 4. Editor in Chief - Philip A. Wilsey, University of Cincinnati. pp. 24-27.
- PXSL Cagle, K. et al. "Professional XSL". Wrox Press Ltd. UK. 2001. ISBN: 1-861003-57-9.
- RDCO Conrad, A. & Obasanjo, D. "XML & Relational Databases". *Dr. Dobb's Journal. Algorithms*. Volume 28, Issue 5. May 2003. pp. 4.
- RLVM Rein, Lisa. "Vector Markup Language". 1998. Available Online at: <http://www.xml.com/lpt/a/98/06/vector/vmlmain.html>

- RPTT Taylor, ES. "An Interim Report on Engineering Design". Massachusetts Institute of Technology. Cambridge, MA. 1959. Pressman. Available Online at: <http://www.it.lut.fi/opetus/99-00/010758000/Lectures/lecture7.html>
- SALTC "SALT – Speech Application Language Tags (SALT) 1.0 Specification". Cisco Systems Inc., Comverse Inc., Intel Corporation, Microsoft Corporation, Philips Electronics N.V., SpeechWorks International Inc. 15 July 2002. Email: info@saltforum.org. Available Online at: <http://www.saltforum.org>
- SAXH Haifeng, Jiang et al. "Path Materialisation Revisited: An Efficient Storage Model for XML Data". *ACM International Conference Proceedings Series. Proceedings of the thirteenth Conference on Database Technologies – Volume 5*. January 2002. Australia. Volume 24, Issue 2. Australian Computer Society, Inc. ISBN~ISSN: 1445-1336, 0-909925-83-6.
- SAXI Idris, Nazmul. "XML, Java, Databases and The Web". 20th June 1999. Available Online at: <http://developerlife.com/dbsourceintro/default.htm#57103>
- SCMI Srinivasan, S. & Brown, E. "Is Speech Recognition Becoming Mainstream?". *IEEE, Computer*. Vol. 35, No. 4. April 2002. pp. 38-41.
- SDPA Wayne, Rick. "Do-It-Yourself Products". *Software Development*. The People, Products and Practices of Corporate Development. August 2001. Vol. 9, No. 8. pp. 20.
- SDPN Hejlsberg, Anders. "Web Services Major Vendors". *Software Development*. The People, Products and Practices of Corporate Development. November 2001. Vol. 9, No. 11. pp.17.
- SDPPO Minnick, Chris. "An XML Toolkit". *Software Development*. The People, Products and Practices of Corporate Development. October 1999. Vol. 7, No. 10. pp. 46-50.
- SDPS From Bill Gates to Developer & IT Professional. *Software Development*. The People, Products and Practices of Corporate Development. September 2001. Vol. 9, No. 9. pp. 33.
- SEIS Sommerville, Ian. "Software Engineering". Sixth Edition. Addison-Wesley Publishing Company. 2001. UK. ISBN: 0-201-39815-X.
- SMLC "Speech Markup Language" February 1999. Available Online at: <http://www.alphaworks.ibm.com/aw.nsf/0/07DBE052560D487C8825671B00680B25?OpenDocument>,

- SMLW Coyle, Frank. "The Wireless Web". 2001. Available Online at: http://www.informit.com/isapi/product_id~%7BE3C85FF9-A92E-4C60-8A8B-1EB33DDFA67E%7D/element_id~%7B6E458633-7F2B-4100-AD67-DAFFF44DB5A4%7D/st~%7B5403B8D0-9308-4B84-9F63-8040DAD445DF%7D/content/articlex.asp
- SMND Ambler, Scott W. "User Interface Design: Tips and Techniques". Excerpts from *The Object Primer 2nd Edition, Building Object Applications That Work, and Process Patterns*. Cambridge University Press. 2000. Available Online at: <http://www.ambysoft.com/userInterfaceDesign.pdf>
- SOPS Strahl, Rick. "Using Microsoft's SOAP Toolkit for remote object access". West Wind Technologies. 2001. Available Online at: <http://www.west-wind.com/presentations/soap/>
- STGM Myers, Glenford J. "The Art Of Software Testing". John Wiley & Sons, Inc. 1979. USA. ISBN: 0-471-04328-1.
- SVG V Vaughan-Nichols, S. "Will Vector Graphics Finally Make It On The Web?". *IEEE, Computer*. December 2001. Volume 34 Number 12.
- SWDH Hohpe, Gregor. "XML Abuse". *Software Development*. Vol. 10, No. 12. December 2002. pp. 38-39.
- SWEP Pressman, Roger S. "Software Engineering – A Practitioner's Approach". Fourth Edition. Mc Graw – Hill. 1997. USA. ISBN: 0-07-052182-4.
- SWMH Humphrey, Watts S. "Managing the Software Process". Addison-Wesley Publishing Company. 1989. USA. ISBN: 0-201-18095-2.
- TMCL Lerner, Reuven M. "At the Forge: Server-Side Java with Jakarta-Tomcat". *Linux Journal*. Volume 2001, Issue 84es. April 2001. Publisher: Specialised Systems Consultants, Inc. USA. Article No. 10. ISSN: 1075-3583.
- UPI Donley, H. Edward. "XML – Overcoming HTML's Limitations". 28th September 2000. Indiana University of Pennsylvania. Available Online at: <http://www.sdc.iup.edu/outreach/fall2000/xml1/index.html>
- VCMI Danielsen, Peter, J. "The Promise of a Voice-Enabled Web". *IEEE, Computer*. Vol. 33, No. 8. August 2000. pp. 104-106.
- VDIH Houlding, David. "VoiceXML and The Voice-Driven Internet – A more natural interface to the Internet". *Dr. Dobb's Journal*. April 2001. Available Online at: <http://www.ddj.com/print/>,

- VMLI IEEE-Industry Standards and Technology Organisation. 2001. Available Online at: <http://www.voicexml.org/tutorials>,
- WMEI Weiser, Mark. "An Embedded, Invisible, Every-Citizen Interface". *Xerox Palo Alto Research Centre. Position Papers, On Interface Specifics*. Available Online at: <http://www.nap.edu/readingroom/books/screen/11.html#nomad>
- WPIV Wu, Peter. "An Introduction to XML". 1998. Microsoft. Available Online at: <http://www.infoloom.com/gcaconfs/WEB/granada99/wu.HTM>
- XCMI Wiggins, Richard. "XML: An Interview with Peter Flynn". *IEEE, Computer*. Vol. 33, No. 4. April 2000. pp. 113-115.
- XDBI Seligman L. & Rosenthal A. "XML's Impact on Databases and Data Sharing". *IEEE, Computer*. Vol. 34, No. 6. June 2001. pp. 59-67.
- XDBO Obasanjo, Dare. "An Exploration of XML In Database Management Systems". 2001. Available Online at: <http://www.25hoursaday.com/StoringAndQueryingXML.html>
- XEBB Bosak, Jon. "The Role of XML in eBusiness". *ebXML Information Day*. Vienna. 9th May 2001. Sun Microsystems.
- XEGP Pierce, John. "XML By Example". QUE. USA. December 1999. ISBN: 0-7897-2242-9.
- XJAV Morgenthal, JP. *Director of Research, NC. Focus*. "Portable Data/Portable Code: XML & Java Technologies". Sun Microsystems, Inc. *White Paper*. Contact: jp@ncfocus.com. Available Online at: <http://java.sun.com/xml/ncfocus.html>
- XJVM Moraes, Ian. "VoiceXML, CCXML and SALT". *XML Journal*. Volume 3, Issue 9. September 2002. Available Online at: <http://www.sys-con.com/xml/article.cfm?id=489>. Email the author at: IMoraes@yahoo.com
- XMDH Hildreth, Suzanne. "XML Marks the Data". October 1998. Available Online at: <http://webserver.cpg.com/features/f1/3.10/>,
- XMLH Harold, Elliotte Rusty. "XML Bible". 2nd Edition. Hungry Minds Inc. USA. 2001. ISBN: 0-7645-4760-7.
- XMLN Harold, Elliotte Rusty & Means, W Scott. "XML In A Nutshell – A Desktop Quick Reference". O'Reilly. 2nd Edition. June 2002. Editor: Simon St. Laurent. USA. ISBN: 0-596-00292-0.

- XMLU Morrison, Michael, et al. "XML Unleashed - From Knowledge to Mastery". SAMS. 1st Edition. December 21, 1999. ISBN: 0-672-31514-9.
- XSLB Burke, Eric M. "Java and XSLT". O'Reilly. September 2001. Editor: Mike Loukides. 1st Edition. ISBN: 0-596-00143-6.
- XSLTI "The XML Culture Clash". Editor: Michael J. Lutz. *IEEE, Computer, Wire Less*. Vol. 35, No. 10. October 2002. pp. 72.
- XTUT XML Tutorial. *White Paper*. Microsoft Corporation. November 2000. Cited at: <http://www.itpapers.com/cgi/PsummaryIT.pl?paperid=20084&scid=201>

Appendix B

Bibliography

The following is a suggested reading list for those wishing to enhance their knowledge, which was found to be beneficial in the area of XML:

- 1 Lucas, Bruce. "VoiceXML for Web-Based Distributed Conversational Applications". *Communications of the ACM*. Volume 43, Issue 9. ACM Press, New York. September 2000. ISSN – 0001-0782. Pages 53-57.
- 2 "XSLT Powers a New Wave of Web Applications". *Linux Journal*. Volume 2002, Issue 95. Specialised Systems Consultants, Inc. USA. ISSN 1075-3583.
- 3 Ceponkus, A. & Hoodbhoy, F. "Applied XML - A Toolkit for Programmers". WILEY. USA. Book & CD Edition. July 1, 1999. 0-4713-44028.
- 4 Fitzgerald, Michael. "Building B2B Applications with XML - A Resource Guide". WILEY. Canada. 2001. 0-471-40401-2.
- 5 St. Laurent, Simon & Cerami, Ethan. "Building XML Applications". USA. 1999. McGraw-Hill. 0-07-134116-1.
- 6 Hunter, David et al. "Beginning XML". 2nd Edition. WROX. USA. 2000. 1-861003-4-12.
- 7 Butler Group. "Database Management Systems – Managing Relational and XML Data Structures". *Technology Evaluation and Comparison Report*. September 2002. Volume 3. ISBN 0-9542845-2-6. UK.
- 8 Berg, Clifford J. "Advanced Java 2. Development for Enterprise Applications". 2nd Edition. Sun Microsystems Press. Prentice Hall. ISBN 0-13-084875-1. USA. 2000.
- 9 McMullin, Barry. "Users with Disability Need Not Apply? Web Accessibility in Ireland". *First Monday*. Volume 7 Number 12. December 2002. Available online at: <http://eaccess.rince.ie>
- 10 Graves, Mark. "Designing XML Databases". Prentice Hall Inc. 2002. USA. ISBN: 0-13-088901-6.
- 11 Reese, George. "Database Programming with JDBC and Java". 1st Edition. O'Reilly. June 1997. Editor: Andy Oram. USA. ISBN: 1-56592-270-0.
- 12 Malhotra, Naresh K. "Market Research, An Applied Orientation". Third Edition. Prentice Hall. 1999. USA. ISBN: 0-13-083044-5.
- 13 Gennick, Jonathan et al. "Oracle8i DBA Bible". IDG Books Worldwide. 2000. USA. ISBN: 00-7645-4623-6.
- 14 Williams, Kevin et al. "Professional XML Databases". Wrox Press Ltd. 2000. Published in UK. Printed in Canada. ISBN: 1-861003-58-7.
- 15 "XML's Impact on Databases and Data Sharing". *IEEE, Computer*. Vol. 34, No. 6. June 2001.
- 16 Pierce, John. "XML By Example". QUE. USA. December 1999. 0-7897-2242-9.
- 17 Harold, Elliotte Rusty. "XML Bible". 2nd Edition. Hungry Minds Inc. USA. 2001. ISBN: 0-7645-4760-7.
- 18 Harold, Elliotte Rusty & Means, W Scott. "XML In A Nutshell – A Desktop Quick Reference". O'Reilly. 2nd Edition. June 2002. Editor: Simon St. Laurent. USA. ISBN: 0-596-00292-0.
- 19 Morrison, Michael, et al. "XML Unleashed - From Knowledge to Mastery". SAMS. 1st Edition. December 21, 1999. 0-672-31514-9.
- 20 Burke, Eric M. "Java and XSLT". O'Reilly. September 2001. Editor: Mike Loukides. 1st Edition. ISBN: 0-596-00143-6.

- 21 Harold, Elliot Rusty. "Processing XML with Java: A Guide to SAX, DOM, JDOM, JAXP, and TrAX". Addison Wesley Professional. 1st Edition. November 2002. ISBN: 0201771861.

lyit | Institiúid Teicneolaíochta Leitir Ceanaínn
Letterkenny Institute of Technology

Appendix C

Glossary/Nomenclature

lyit | Institiúid Teicneolaíochta Leictir Ceanalinn
Letterkenny Institute of Technology

Glossary/Nomenclature

This glossary gathers and defines the terms and abbreviations utilised in this thesis. Terms in the glossary are defined under their acronyms.

A

Applet

An applet is a program written in the Java programming language that can be embedded an HTML page using the <applet> tag.

Application Programme Interface (API)

An API is a series of functions that programs can use to make the operating system do most of the work. Using Windows APIs, for example, a program can open windows, files, and message boxes, as well as perform more complicated tasks by passing a single instruction.

American Standard Code for Information Interchange (ASCII)

American Standard Code for Information Interchange - It is the basis of character sets used in almost all present-day computers.

Attributes

An attribute is information that relates to an entity. For example name and address are attributes of a person.

AVI

Audio Video Interleave - A video file format.

B

Bitmap

A bitmapped graphic contains a list of colours of individual pixels in a normally rectangular area. Bitmapped graphics' resolution is set for a certain image-map size and as a result of this, it distorts when scaled to work with a small display or a high-resolution printer [GSYL].

Bits

Is a fundamental unit of information having just two possible values, as either of the binary digits 0, or 1.

Browser

A program specifically designed to help users view and navigate hypertext, on-line documentation, or a database.

Business to Business (B2B)

B2B generally involves a relationship between the supplier and a purchasing company (in other words between two businesses).

Business to Consumer (B2C)

B2C generally involves a relationship between the supplier and a consumer (in other words between a business and a consumer).

Bytes

A sequence of bits, usually eight, operated on as a unit by a computer.

C**C**

A procedural programming language widely used for systems programming.

C++

An object oriented programming language widely used for systems programming.

Cache

A small fast memory holding recently accessed data, designed to speed up subsequent access to the same data. Most often applied to processor-memory access but also used for a local copy of data accessible over a network [GSYL].

Cascading Style Sheet (CSS)

The CSS can be applied to an HTML or an XML file to format a web page so that the HTML code can be less burdened with formatting and more focused on content.

Central Processing Unit (CPU)

That part of a computer that interprets and executes instructions.

Class Data (CDATA)

Defines behaviour in the DOM.

Client

A computer or program that can download files for manipulation, run applications, or request application-based services from a file server.

Compression

The process by which data is compressed into a form that minimises the space required to store or transmit it [GSYL].

Corba

Common Object Request Broker Architecture - specification, which provides the standard interface definition between OMG-compliant objects [GSYL].

D**Data Islands**

A data island is an XML document that exists within an HTML page. It allows the developer to script against the XML document without having to load it through script or through the <OBJECT> tag.

Data-Centric

A model where data is stored in a relational database or similar repository.

Distributed Component Object Model (DCOM)

Distributed Component Object Model - Microsoft's extension of their Component Object Model (COM) to support objects distributed across a network.

Document Object Model (DOM)

A W3C specification for APIs for accessing the content of HTML and XML documents.

Document Type Definition (DTD)

The DTD defines valid tags and attributes, and their allowed contents.

E**Electronic Data Interchange (EDI)**

Electronic Data Interchange - The exchange of standardised document forms between computer systems for business use. It can also be described as the electronic exchange of formatted data.

Enterprise Java Beans (EJB)

Enterprise Java Beans – This is a server-side component architecture for writing reusable business logic and portable enterprise applications [GSYL].

Electronic Commerce (E-Commerce)

The conducting of business communication and transactions over networks and through computers. E-Commerce could also be defined as the buying and selling of goods and services, and the transfer of funds, through digital communications.

Element

Descriptive content containers are called elements and are often referred to as tags. Elements are usually given clear English-based names that are based on their function or content, such as PARAGRAPH, TABLE etc.

Encryption

Any procedure used in cryptography to convert plaintext into ciphertext in order to prevent any but the intended recipient from reading that data.

Entity

Something that exists as a particular and discrete unit and can be considered to be the existence of something considered apart from its properties.

Extensible Hypertext Markup Language (XHTML)

A reformulation of HTML 4.01 in XML. With the incorporation of XML, that means that XHTML can be viewed, edited, and validated with standard XML tools.

Extensible Markup Language (XML)

XML is a markup language for documents incorporating structured information. Structured information contains both content (words and pictures) and an indication of the functionality of the content.

Extensible Path (XPath)

XPath is string syntax for constructing addresses to the information found in an XML document. This language is used to specify the locations of document structures or data found in an XML document when processing that information using XSLT.

Extensible Stylesheet Language (XSL)

The browser translates the XML data into HTML using the XSL document. This way, only the essential data is passed through the network multiple times.

Extensible Stylesheet Language Transformation (XSLT)

XSLT is used to specify how an implementation of an XSLT processor is implemented to create the preferred output from a given marked-up input. Therefore, XSLT enables interoperability. By using XSLT it is possible to turn an XML document into HTML, PDF or another form of XML

F**Forms [] Array**

Using the forms[] array each Form object can be accessed in turn. It has a length property, which is required to see how often a function needs to be looped through.

Fourth Generation Language

An "application specific" language. The term was invented by Jim Martin to refer to non-procedural high level languages built around database systems, e.g. SQL [GSYL].

G**Graphics Interchange Format (GIF)**

Graphics Interchange Format - A service mark used for a raster-based colour graphics file format, often used on the World Wide Web to store graphics.

Graphical User Interface (GUI)

An interface for issuing commands to a computer utilising a pointing device, such as a mouse and/or a keyboard that manipulates and activates graphical images on a monitor.

H**Hyper Text**

It is a computer-based text retrieval system that enables a user to access particular locations in web pages or other electronic documents by clicking on links within specific web pages or documents.

Hyper Text Markup Language (HTML)

HTML is considered to be the basis for publishing hypertext on the World Wide Web. It is a non-proprietary format based upon the SGML. Refer to Section x in Chapter 2.

Hyper Text Transfer Protocol (HTTP)

A protocol used to request and transmit files, especially web pages and web pages components, over the Internet or other computer network.

I**Inheritance**

In object-oriented programming, the ability to derive new classes from existing classes.

Inline

Can mean that an image is part of the page and is either next to or surrounded by text.

Interface

The point of interaction or communication between a computer and any other entity, such as a printer or human operator.

International Standards Organisation (ISO)

The ISO is an organisation that sets standards in many businesses and technologies, including computing and communications.

Internationalisation

The process and philosophy of making software portable to other locales. For successful localisation, products must be technically and culturally neutral (no language barriers)

Interoperability

Is the ability of software and hardware on multiple machines from multiple vendors to communicate.

Intranet

A network based on TCP/IP protocols belonging to an organisation usually a corporation accessible only by the organisations' members or authorised personnel.

J**Java**

An object-oriented language originally developed at Sun by James Gosling. It is a powerful object oriented trademark used as a programming language designed to develop applications, especially applications for the Internet that can operate on different platforms [GSYL].

Java Development Kit (JDK)

A free Sun Microsystems product, which provides the environment, required for programming in Java. The JDK is available for a variety of platforms, but most notably Sun Solaris and Microsoft Windows.

Java Server Pages (JSP)

A freely available specification for extending the Java Servlet API to generate dynamic web pages on a web server. The JSP specification was written by industry leaders as part of the Java development program. JSP assists developers in creating HTML or XML pages that combine static (fixed) page templates with dynamic content [GSYL].

JavaScript

JavaScript should not be confused with Java. JavaScript is a Sun trademark. JavaScript is intimately tied to the World-Wide Web, and currently runs in only three environments - as a server-side scripting language, as an embedded language in server-parsed HTML, and as an embedded language run in browsers.

Joint Photographic Expert Group (JPEG)

The standard algorithm for the compression of digital images.

K**L****Legacy Database**

As a natural progression, the new data becomes old and the software inevitably becomes difficult to use. As a result legacy data stored in databases is unavoidable.

M**Markup Languages (ML)**

A markup language may be described as a language that provides information about information or meta information.

Mathematica

Mathematica is a markup language that can be used to put mathematics on to the Web.

Mathematical Markup Language (MathML)

MathML is an XML application that provides a low-level method of communicating mathematics between machines. MathML enables mathematical expressions to be displayed, manipulated and shared over the Internet

Metadata

Is information about data. In data processing, meta-data provides information about, or documentation of, other data managed within an application or environment.

Middleware

Middleware is a layer of software between the network and the applications.

MINSE

MINSE was created in 1996, and was one of the first languages employed to help render mathematical notations to the Internet.

Module

A portion of a program that carries out a specific function and may be used alone or combined with other modules of the same program.

Moving Pictures Expert Group (MPEG)

Moving Pictures Expert Group - Any of a set of standards established for the compression of digital video and audio data.

N**Node**

The Node interface defines properties and methods for navigating and manipulating a tree representation of a document.

O**Object**

A discrete item that can be selected and manoeuvred, such as an onscreen graphic. In object-oriented programming, objects include data and the procedures necessary to operate on that data [GSYL].

Ontology

A means of exchanging structured information. Ontology is now accepted as a fundamental building block of Business-to-Business (B2B), Business -to-Consumer (B2C) Communication and E-Commerce.

P**Parser**

The parser used to obtain information is simply a program that can read XML syntax. By using a parser the finished application will never have to look at the XML for the information, thus speeding up the process.

Personal Digital Assistant (PDA)

Personal Digital Assistant - A lightweight, hand-held, usually pen-based computer used as a personal organiser.

Perl

A high-level programming language developed by Larry Wall in 1987 and developed as an open source project [GSYL].

Pixel

The basic unit of the composition of an image on a television screen, computer monitor, or similar display. The greater the number of pixels per inch the greater the resolution.

Platform Independent

The Platform is the basic technology of a computer system's hardware and software that defines how a computer is operated and determines what other kinds of software can be used. If an application is 'independent', then these factors do not play a very influential role [GSYL].

Plugin

A file containing data used to alter, enhance, or extend the operation of a parent application program. Plugins can usually be downloaded for free and can be incorporated by using the <EMBED> tag within an HTML document.

Portable Network Graphics (PNG)

Portable Network Graphics - An extensible file format for the lossless, portable, well-compressed storage of raster images.

Q**Queries**

A user's request for information, generally as a formal request to a database or search engine. SQL is the most common database query language.

R**Random Access Memory (RAM)**

A memory device in which information can be accessed in any order.

Relational Database

A database system in which any database file can be a component of more than one of the database's tables.

Resolution

The maximum number of pixels that can be displayed on a monitor, expressed as (number of horizontal pixels) multiplied by (number of vertical pixels), i.e., 1024x768. The ratio of horizontal to vertical resolution is usually 4:3, the same as that of conventional television sets.

S**Scalable Profile**

A scalable profile enables users of varying programming abilities with varying technologies to create SMIL documents that will act intelligently and which are tailored to the capabilities of the target devices.

Scalable Vector Graphics (SVG)

SVG allows Web developers and designers to dynamically create high quality graphics from real-time data with very precise structural and visual management.

Server

A computer that processes requests for HTML and other documents that are components of web pages.

Simple API for XML (SAX)

SAX is a Java technology interface that permits applications to combine with any XML parser to obtain notification of parsing events.

Simple Object Access Protocol (SOAP)

SOAP is an XML based protocol used for exchanging information in a decentralised, distributed environment.

Span

This element is used to create a structure in a document. By using this element it is possible to give part of the document a name, or apply style sheet information to the part.

Speech Application Language Tags (SALT)

SALT is comprised of a small set of XML elements, with associated attributes and DOM object properties, events and methods, which may be used in conjunction with a source markup document to apply a speech interface to the source page.

Speech Markup Language (SpeechML)

SpeechML provides tags for defining spoken output and input as well as tags for initialising actions to be taken on a given spoken input.

Standardised General Markup Language (SGML)

SGML is a formal system designed for building text markup languages.

Stroke

The stroke colour is used to frame shapes and lines with colour.

Synchronised Multimedia Integration Language (SMIL)

SMIL is a markup language designed to be easy to learn and install on Web sites. SMIL was created specifically to solve the problems of coordinating the display of multimedia on Web sites.

T**Tags**

HTML tags are usually English words or abbreviations but they are distinguished from the normal text because they are placed in small angle brackets. They essentially tell the browser what to do.

Text To Speech (TTS)

A means of converting text to speech. This technology is used in VoiceXML.

Tomcat

Tomcat is a servlet container and is a means for implementing JavaServer Pages. It may be used stand alone, or in conjunction with a web server.

Transfer Control Protocol/Internet Protocol (TCP/IP)

A protocol for communication between computers, used as a standard for transmitting data over networks and as the basis for standard Internet protocols.

U**Unicode**

Unicode is the World's standard for encoding text.

Universal Resource Identifier (URI)

Universal Resource Identifier - The generic set of all names and addresses, which are short strings, which refer to objects on the Internet [GSYL].

Uniform Resource Locator (URL)

Uniform Resource Locator - An Internet address (for example, *http://www.hmco.com/trade/*), usually consisting of the access protocol (*http*), the domain name (*www.hmco.com*), and optionally the path to a file or resource residing on that server (*trade*) [GSYL].

Uniform Resource Name (URN)

Uniform Resource Name – It is essentially any URI, which is not a URL. For example: *urn:hdl:cnri.dlib/august03*

V**Vector**

The representation of separate shapes such as lines, polygons and text, and groups of such objects.

Vector Markup Language (VML)

VML supports vector graphic information in the same way that HTML supports textual information and has been available since 1998.

Voice Extensible Markup Language (VoiceXML)

VoiceXML is a Web-based markup language for representing human-computer dialogues, just like HTML.

Voice Interpreter

In VoiceXML the user interacts with the system by relating a message via a telephone for example. The message is then recognised by the speech recognition and is then sent to the Voice Interpreter. If the grammar is recognised, then a synthesised response is given to the user.

W**WAV**

A sound format developed by Microsoft and used extensively in Microsoft Windows.

What You See Is What You Get (WYSIWYG)

Describes a user interface for a document preparation system under which changes are represented by displaying a more-or-less accurate image of the way the document will finally appear.

World Wide Web (WWW)

An Internet client-server hypertext distributed information retrieval system, which originated from the CERN High-Energy Physics laboratories in Geneva, Switzerland [GSYL].

World Wide Web Consortium (W3C)

The main standards body for the World-Wide Web. W3C works with the global community to establish international standards for client and server protocols that enable on-line commerce and communications on the Internet. It also produces reference software.

X**XML Namespace**

An XML namespace is a mechanism to identify XML elements. A namespace is a solution to help manage XML extensibility.

XML Schema

A schema is any type of model document that defines the structure of something.

XML-Dev

XML-Dev is a mailing list, which is under the direction of David Megginson and is used for the productive discussion of XML topics.

Appendix D

Acronyms

Acronyms

API	Application Programming Interface
ASP	Active Server Page
AVI	Audio Video Interleave
B2B	Business-To-Business
B2C	Business-To-Consumer
CDATA	Class Data
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CSS	Cascading Style Sheet
DCOM	Distributed Component Object Model
DFD	Data Flow Diagram
DOM	Document Object Model
DTD	Document Type Definition
EDI	Electronic Data Interchange
EJB	Enterprise Java Beans
ERD	Entity Relationship Diagram
GIF	Graphics Interchange Format
GML	General Markup Language
GUI	Graphical User Interface
HCI	Human Computer Interaction
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
ISO	International Standards Organisation
JDK	Java Development Kit
JPEG	Joint Photographic Expert Group
JSP	Java Server Pages
MathML	Mathematical Markup Language
MPEG	Moving Pictures Expert Group
PDA	Personal Digital Assistant
PNG	Portable Network Graphics
RAM	Random Access Memory
SALT	Speech Application Language Tags
SAX	Simple API for XML
SGML	Standardised General Markup Language
SMIL	Synchronised Multimedia Integration Language
SOAP	Simple Object Access Protocol
SpeechML	Speech Markup Language
SVG	Scalable Vector Graphics
TCP/IP	Transfer Control Protocol/Internet Protocol
TTS	Text To Speech
UML	Unified Modelling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Number




VML	Vector Markup Language
VoiceXML	Voice Extensible Markup Language
W3C	World Wide Web Consortium
WWW	World Wide Web
WYSIWYG	What You See Is What You Get
XHTML	Extensible Hyper Text Markup Language
XLink	Extensible Link
XML	Extensible Markup Language
XPath	Extensible Path
XSL	Extensible Stylesheet Language
XSLT	Extensible Stylesheet Language Transformation

Appendix E





Code Keys

 Code Keys


UML - Use Case Notation

<u>Notation</u>	<u>Explanation</u>
	Actor: A person, an organisation, a system, or some other external entity, which must interact with the system.
	Use-case: A business function – Actors interact with a use case in order to accomplish a task or activity.
	Interaction: Interaction arrow shows where an actor interacts with a use case.

UML - Dynamic Modelling Notation

<u>Notation</u>	<u>Explanation</u>
	Start Symbol: Identifies the beginning of the life cycle of the class and starting point for the diagram. It precedes the initial creation of the class.
	State: A distinct condition that a class may be in during its lifetime.
	Transition: Indicates a change from one state to another.
	End Point: Indicates the end of the class' life cycles. It follows the deletion or destruction of the class.

UML – Business Process Modelling Notation

<u>Notation</u>	<u>Explanation</u>
	Process: This represents what the system will do. It is essentially the software that will be developed.



Input: This is representative of the problem definition to be developed.



Resources/Goals/Outputs: The box can represent what will be used in the development of the software (i.e. resources) for example an SQL Server or a web page. It also represents the goals of the system, for example to query a database. Finally, it represents the output of the system.



Line: Connects the process to its resources.

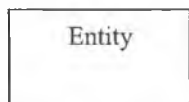


Arrow: Indicates the flow of information into and from the process.

Entity Relationship Diagrams

Notation

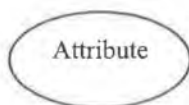
Explanation



Entity: An entity is something about which information is stored, or an action is carried out on.



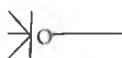
Action: An action is something that is carried out on an entity, e.g. 'created'



An attribute is something that is associated with an entity. For example an entity might be a 'person' and the attributes of that 'person' might be 'name', 'address', etc.



Relationship: This represents a 'One' relationship



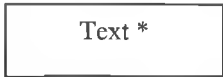



Relationship: This represents a 'Zero or more, Optional' relationship.







Relationship: This represents a 'Many' relationship.

Jackson Structure Diagram

<u>Notation</u>	<u>Explanation</u>
	Sequence: Is represented by unmarked boxes containing a description.
	Lines: Connect the sequences in the diagram.
	Iteration: Is represented by an asterisk in the box. It implies that the system user will have a repetition occurring.
	Selection: Is represented by a circle in the box. It implies that the system user will be able to make a selection.

Data Flow Diagrams

<u>Notation</u>	<u>Explanation</u>
	Entity: An entity tends to either produce or consume information that remains outside the bounds of the system to be modelled.
	Process: This transforms that information that resides within the system.
	Data Flow: Depicts the flow of data in the diagram.
	Data Store: This represents data that is to be stored for use by one or more of the processes.

Appendix F

Questionnaire/Test Data

Instructions



Please follow the instructions carefully:

- 1. Read the 'Questionnaire' sheet first.*
- 2. Read the 'Assignment Sheet' second.*
- 3. Ask whoever is conducting the evaluation any questions that you are unsure of before commencing the tasks.*
- 4. You may use the help (available on the "Home Page") and/or ask questions as necessary throughout the duration of the evaluation.*
- 5. You may spend as long as you wish reading the 'Questionnaire' and 'Assignment Sheet' before commencing to complete the tasks.*
- 6. However, only 10 minutes will be allocated to the assignment itself (i.e. carrying out the tasks). Make sure to use the headset provided!*
- 7. When the 10 minutes have elapsed, the person conducting the evaluation will inform you and you will then be asked to complete the 'Questionnaire'. You may take as long as you wish to complete the 'Questionnaire'.*

Thank you for taking the time to fill this out.

Assignment Sheet



Ensure that you have read the 'Instructions' sheet and the 'Questionnaire' sheet before carrying out the tasks.

Thank you for taking the time to complete this short assignment and questionnaire. This assignment will take 10 minutes to complete and its purpose is to help evaluate software that was created using XML and applications of XML (i.e. SVG (Home Page etc), VML (Help Pages), SMIL and MathML).

Please read all the relevant tutorials before answering any questions.

Attempt as many of the questions as you can within the time allocated.

Thank you.

Note:

- Where a “Home Page” link is not provided please use the ‘Back’ button located at the top left-hand corner of your screen.
- Also, when you are asked to read the on-line tutorial first, you may do this by selecting the “SVG Page” from the “Home Page” or by clicking on the “Alternative Tutorials” link located on the “Home Page”.
- Please read the “Help” page on the “Home Page” as an introduction.

Step 1: Click on the links to “Enter the Home Page”.

Step 2: Select the button at the right-hand side of the screen labelled “SVG Page”.

Task 1 **Read all of the information before answering a question!*

Question 1: What is the main advantage of SVG on the Internet?

Answer:

Step 3: Return to the “Home Page” and select the “Alternative Tutorials”.

Step 4: Select the ‘Tutorial on SVG through XML’.

Task 2 **Read all of the text before answering the question!*

Question 2: Who has taken a leadership role in developing SVG?

Answer:

Step 5: Return to the “Home Page” by hitting the ‘Back’ button and selecting the “Home Page” link below the menu.

Step 6: Click on the “SMIL” button located at the bottom right-hand corner.

Step 7: Click on the square labelled “Steps”.

Task 3 **Listen carefully and read the text too!*

Question 3: What kind of language is SMIL?

Answer:

Question 4: Where can SMIL be used?

Answer:

Step 8: Click on the middle square to view the 'Video'.

Question 5: What format was the video converted into before it could be embedded into the SMIL presentation?

Answer:

Step 9: Return to the "Home Page" and go to the "Tutorial Summary".

Step 10: Follow the link for an 'Interactive XML Tutorial' located under the date.

Step 11: 'Add' the product and 'Create the XML Code'.

Step 12: Click on the button "Format the XML for Display" to view the code.

Task 4

Question 6: If an XML document had an opening tag <name>, how would you close this tag?

Answer:

Step 13: Return to the "Home Page".

Step 14: Return to the “Tutorial Summary” and select the “Tutorial on SVG.xml” located under the date.

Task 5

Question 8: After reading the explanations for each line of code given (i.e. Line 1.xml etc), which line(s) specifies the grammar?

Answer:

Note: Use the ‘Back’ button to return to the previous page (“Tutorial Summary”), then hit the ‘Back’ button again to return to the “Home Page”.

Question 9: How would you end an SVG document? *Hint: a tag! If you need to return to the tutorial, please do.*

Answer:

Step 15: Return to the “Home Page” by hitting the ‘Back’ button (if you’re not already there) and ‘Look at the MS Access Database’.

Step 16: Look at the ‘Steps’ involved on the SMIL presentation located at the bottom of the screen.

Task 6

Question 10: What does the executeQuery() function return?

Answer:

Step 17: Look at the ‘Video’.

Question 11: Give one advantage to displaying data from a database in XML format?

Answer:

Questionnaire



Please read this questionnaire before carrying out the tasks.

This questionnaire is aimed at determining the advantages/disadvantages offered by the piece of software to be tested, therefore, it is very important that you answer the questions to the best of your ability otherwise the results will be inaccurate.

Thank you for taking the time to fill this out.

Gender

Male Female

Have you ever heard of XML?

Yes No

Level of computer literacy

Experienced Novice Non-Computer User

Age Bracket

<18 18-24 25-34 35-44 45-54 >55

Please indicate the level of agreement/disagreement with the following statements using the scale below.

1. On the second screen entitled “XML Technologies In Use”, the graphics were visually pleasing.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

2. On the “Home Page”, the interface was easy to use.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

3. The text was easy to read on the “Home Page”.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

4. The animation was not visually pleasing on the “Home Page”.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

5. In the presentation on SMIL, the user interface was easy to use.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

6. The 'slides' on the SMIL presentation were not easy to read.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

7. The sound of the spoken dialogue was of good quality in the SMIL presentation (i.e. no interference/hissing sounds).

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

8. The 'video' was of good quality (i.e. sound matched the image).

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

9. The sound quality on the 'video' was good (i.e. no interference).

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

10. SMIL was not a good idea.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

11. On the “SVG Page”, the text was not easy to read.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

12. The interface was visually pleasing.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

13. On the “Alternative Tutorials” page, the “Tutorial on SVG” was easy to read.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

14. I preferred the ‘look and feel’ of the “Alternative Tutorials” to the “SVG Page” as it was easier to read.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

15. I did not like the interface on the “Tutorial Summary” page.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

16. The “Interactive XML Tutorial” (located under “Tutorial Summary”) was easy to use.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

17. I learnt something new from this tutorial.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

18. The interface on the “Tutorial on SVG” (within the “Tutorial Summary”) explaining a basic SVG Document Structure was visually pleasing.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

19. I did not learn something new from this tutorial.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

20. Generally, it was easy to move from screen to screen.

Strongly Agree	Agree	Neither Agree/ Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

21. I was able to find my way back to the "Home Page" easily.

Strongly Agree	Agree	Neither Agree/Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

22. There was adequate help available.

Strongly Agree	Agree	Neither Agree/Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

23. XML serves a useful purpose.

Strongly Agree	Agree	Neither Agree/Disagree	Disagree	Strongly Disagree
[]	[]	[]	[]	[]

Please indicate your level of satisfaction with each of the following statements by circling the rating you find most appropriate. The values range from 1 (very poor) to 5 (excellent).

1. Visual Appearance of the:

"Home Page"	1	2	3	4	5
"SVG Page"	1	2	3	4	5
"SMIL" Presentation	1	2	3	4	5
"Interactive XML Tutorial"	1	2	3	4	5
"Help" Pages	1	2	3	4	5
"Tutorial on SVG through XML"	1	2	3	4	5

2. Ease of use of the:

"Home Page"	1	2	3	4	5
"SVG Page"	1	2	3	4	5
"SMIL" Presentation	1	2	3	4	5
"Interactive XML Tutorial"	1	2	3	4	5
"Help" Pages	1	2	3	4	5
"Tutorial on SVG through XML"	1	2	3	4	5

3. The use of colour on the:

“Home Page”	1	2	3	4	5
“SVG Page”	1	2	3	4	5
“SMIL” Presentation	1	2	3	4	5
“Help” Pages	1	2	3	4	5
“Tutorial on SVG through XML”	1	2	3	4	5

4. Level of knowledge gained from the:

“SVG Page”	1	2	3	4	5
“SMIL” Presentation (video)	1	2	3	4	5
“SMIL” Presentation (slides)	1	2	3	4	5
“Interactive XML Tutorial”	1	2	3	4	5
“Help” Pages	1	2	3	4	5
“Tutorial on SVG through XML”	1	2	3	4	5

5. Would you recommend this software?

Yes No

6. Would you consider using this software again?

Yes No

If you have any further comments please feel free to send an E-mail (which you can do anonymously) from the “Home Page” by clicking on the ‘envelope’ graphic. The E-mail facility secures your privacy.

Thank you.

Appendix G

Applications Running

G.1 Introduction

This Appendix presents a visual overview of the systems as developed in order to highlight both the capabilities of XML and its ability or lack thereof to interact with a legacy database. Appendix H shows sample test results from the evaluation of these systems.

G.1.1 Sigma-X System: Tutorial Demonstration/Applications of XML

The system, Sigma-X, illustrated below demonstrates a number of the capabilities of XML. XML's interactions with a legacy database system (in this example, MS Access), is also illustrated.

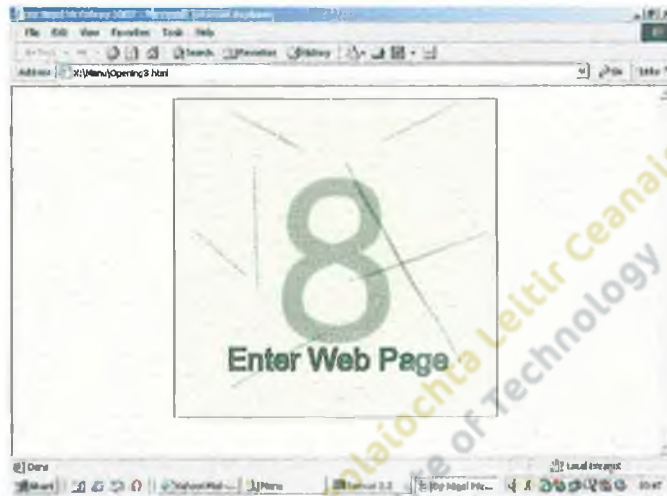


Fig. G.1 Opening Screen of Sigma-X

Figure G.1 shows an SVG page that 'counts down' an introduction to the system. It is similar to the functionality of Flash. A plugin is necessary in order to display SVG.

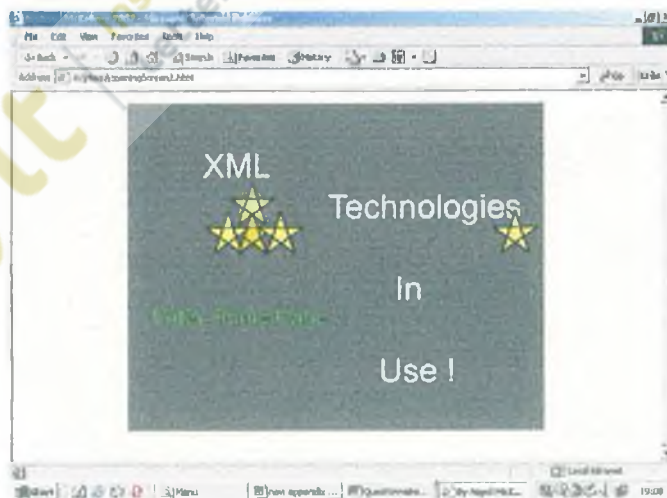


Fig. G.2 Animation in SVG

Figure G.2 provides animation for the user and a point of entry into the main system. The stars move along a specified path which is declared in the code.

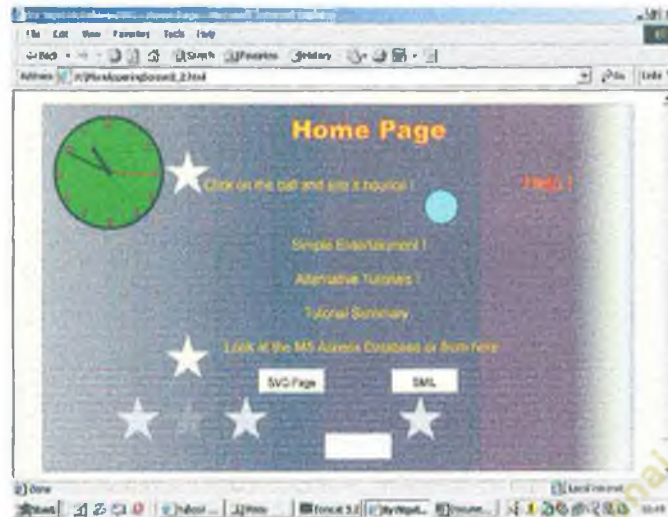


Fig. G.3 Menu options presented in SVG

Figure G.3 displays menu options to the users and allows the user to send an email. Figure G.3 demonstrates how SVG can be interoperable with other languages such as JavaScript. At the outset, it was necessary to determine how XML and applications of XML could interact with other applications. The clock in the top left corner of the screen incorporates both SVG and JavaScript. Figure 3.10 in Chapter 3 also demonstrate how these languages can interact with each other.

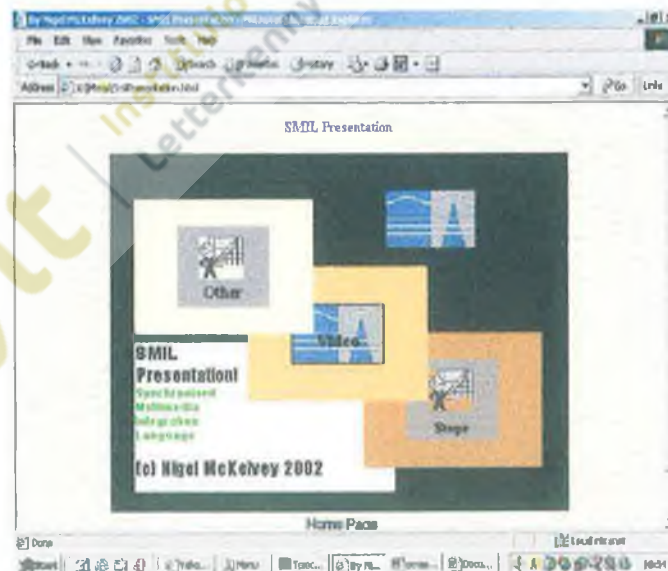


Fig. G.4 SMIL Presentation

Figure G.4 shows a screen shot of the SMIL presentation. Here the user could watch a video as can be seen in Figure G.5 or read through a series of slides as can be seen in Figure G.6. This is embedded within a HTML document using a plugin. An issue, which was raised with SMIL, was that of portability (refer to Appendix J).



Fig. G.5 Video presentation in SMIL



Fig. G.6 Slide presentation in SMIL

The slide presentation is accompanied with audio, which is synchronised with the slides. This is accomplished by specifying timers in the SMIL code. As previously stated (in Chapter 5, Section 5.4), issues were raised about SMIL's capabilities. Particularly as bandwidth can play an important factor is SMIL's ability to present information accurately and correctly without stalling or faltering.



Fig. 6.7 SVG Pages

In Figure G.7 information is presented to the user in an alternative way to HTML. Testing however revealed that users preferred information displayed through XML rather than SVG as it was difficult to read displayed as SVG. This again demonstrates one of the capabilities of XML as set out in Chapter 1. Users can scroll over the menu options and read the 'tool-tip/texts', which were all created in SVG. Users can also email from this page.

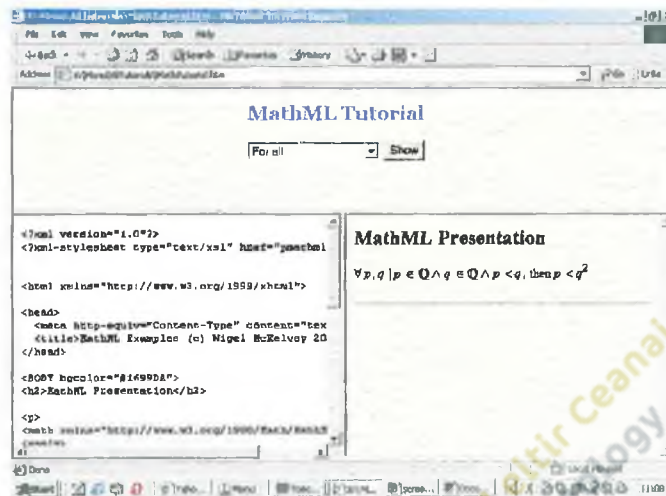


Fig. G.8 MathML Pages

MathML is an application of XML and it allows a developer to present mathematical information to users on a worldwide basis by simply using markup. This eliminates the need to embed large bitmaps or JPEGs into web sites. Again, this demonstrates one of the many capabilities of XML. In the system the markup is presented to the user in the form of a tutorial. Users can select from the drop-down list and view the equation presented on the right and the markup itself on the left.



Fig. G.9 XML as a Database

Figure G.9 demonstrates how an XML document can be used as a database in itself. Here Java was used to retrieve information from the document depending on what the user searches for. This again relates back to Figure 3.10 in Chapter 3 where a diagram depicts the ‘interactions’ of various languages.

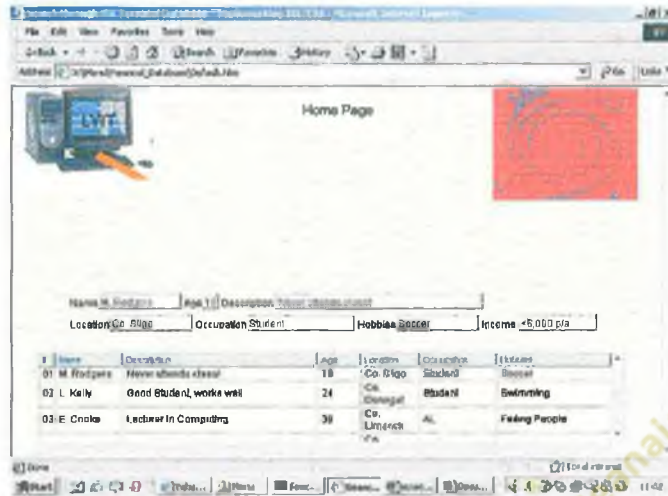


Fig. G.10 Uses of XML - XSL

Figure G.10 depicts the part of the system that demonstrates how a developer can customise his or her own arrowheads, tables, etc. by using XSL and XML Schemas. This screen was simply used as a demonstration as to how XML could be incorporated into a system successfully and how older data stored in relational databases could be extracted, converted into XML files and manipulated into table formats as can be seen above.

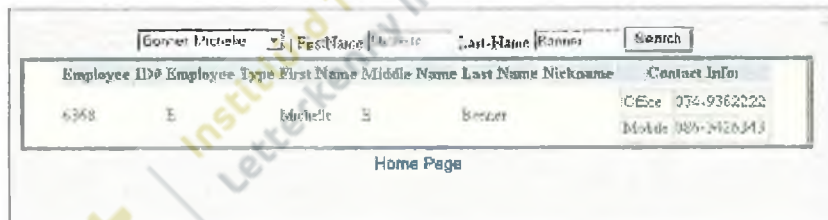


Fig. G.11 Data Islands

Data Islands can hold large quantities of data and embed it in a web page. In the case of XML, the user can then use the document as a database and scroll through and carry out specific searches as necessary as can be seen in Figure G.11.

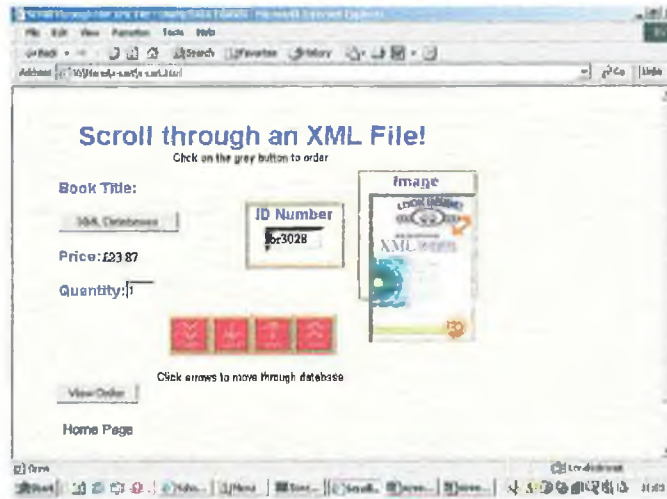


Fig. G.12 XML Implementations

Figure G.12 illustrates how a user can interact with the system by ordering a series of books. The system will automatically total the cost and will allow the user to remove items as desired.

G.1.2 Kappa-C: VoiceXML Capabilities

Figures G.13 and G.14 outline how the Voice XML application was created, depicting the environment that was used (IBM WebSphere) and the application running.

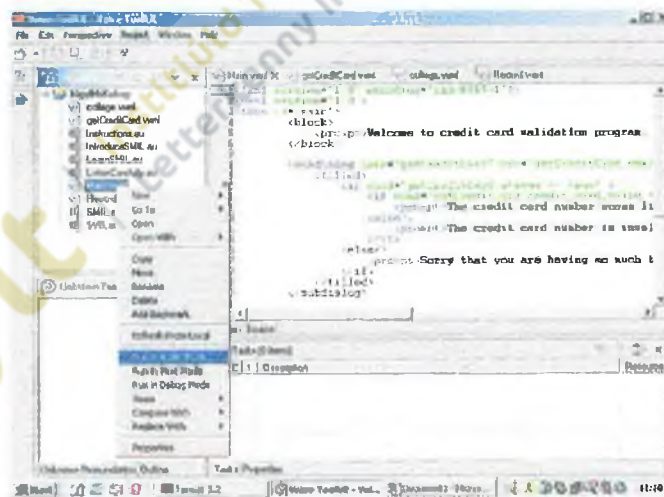


Fig. G.13 VoiceXML Development

Figure G.13 shows the developing environment where the VoiceXML application was developed. Figure G.14 shows the system running. The user can interact with the system by either using a microphone or by using the keyboard. This demonstrates XML's capabilities for improving human computer interactions for individuals with disabilities for example. The user could also interact with the system by using the grey box at the top left hand corner in order to enter in digits.

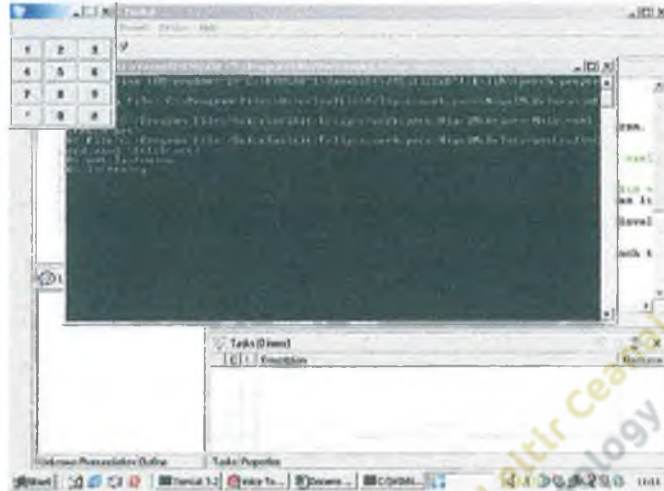


Fig. G.14 VoiceXML Application running

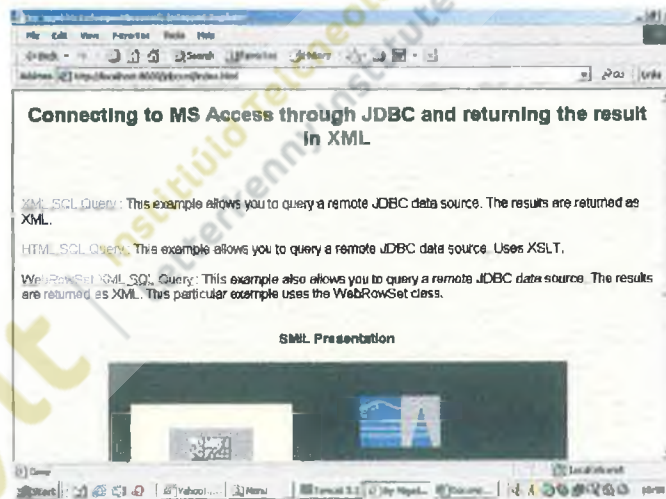


Fig. G.15 XML and MS Access

XML can interact with legacy database systems. Figure G.15 shows the opening screen to the section of the system where the user can 'interact with a legacy database'. As can be seen in Figure G.16 users can enter an SQL statement and click on 'Submit Query'. The system will then retrieve the information and display it as XML as can be seen in Figure G.17. This XML format gives the information more semantic information and can be easily formatted for display in the Internet if desired.

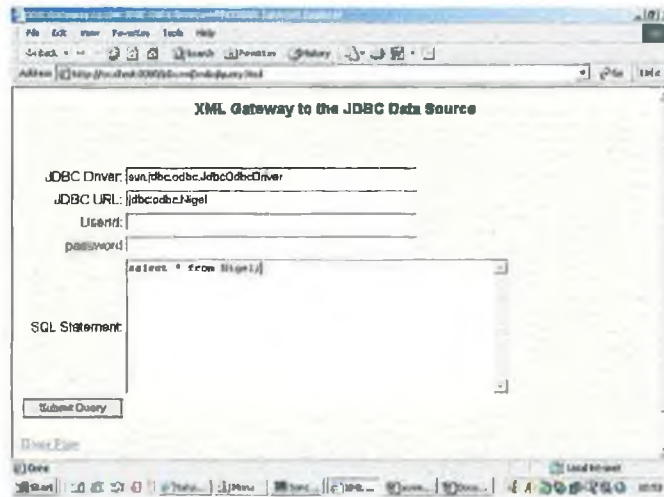


Fig. G.16 Writing the SQL

Figure G.16 shows the SQL statement that will retrieve everything from the database named 'Nigel'. The screen also depicts the JDBC Driver string and the JDBC URL. The user can change this URL in order to retrieve data from a different database.

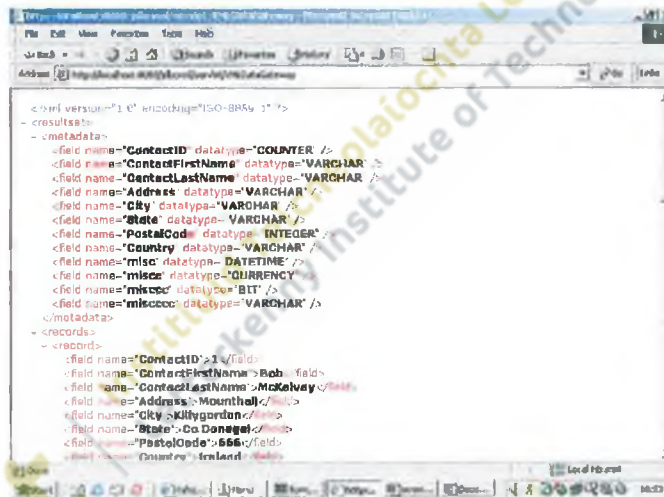


Fig. G.17 Displayed as XML

At the outset it was necessary to determine if it was possible to interact with a legacy database system. This was achieved through the application above. However certain 'returning errors' were detected which is further discussed in Chapter 5, Table 5.3. As can be seen in Figure G.17 above, all of the data in the database had data types, which could be returned without error.

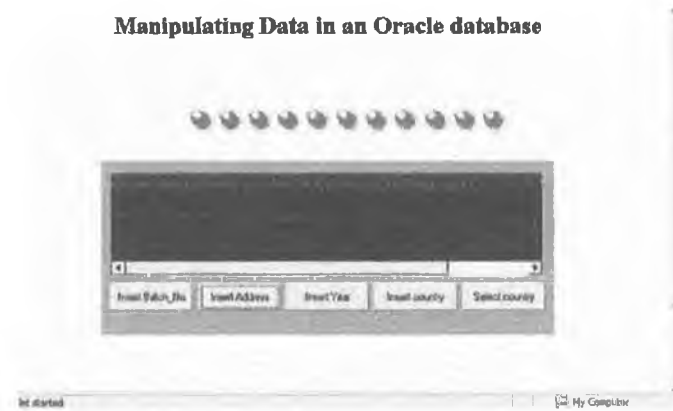
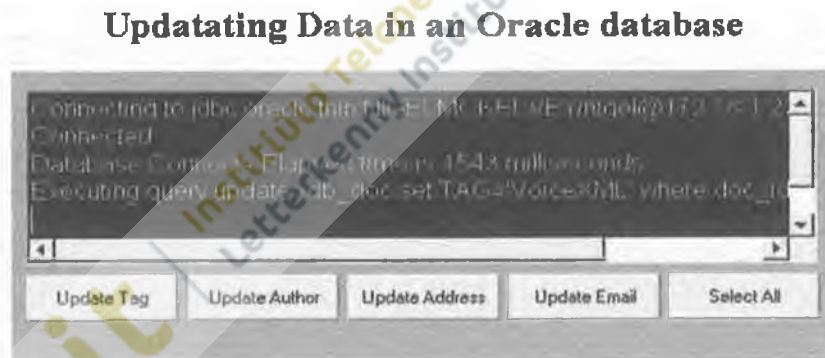


Fig. G.20 Inserting data into the database

Figure G.20 shows how the user can insert new data into the database. In Figure G.20 the 'Insert' option is shown. This screen is repeated for the other operations, including: 'Add', 'Update', 'Select' and 'Delete'. The same screen exists for all of them but the buttons change accordingly with Java code embedded behind each, which executes the appropriate SQL statement. Testing revealed that connection times were relatively quick, however an anomaly was discovered which is discussed in Chapter 5, section 5.9. Figure 5.4 also depicts this anomaly occurring. Appendix H also discusses this problem. Please refer to Figure G.21 for the same example:



Main Page

Fig. G.21 Anomaly occurring in Oracle

The anomaly detected in the Oracle system and the returning errors detected in the MS Access system implies that the 'Problem Definition' established was thoroughly investigated.

Also, the problems encountered with the applications of XML, such as, browser support issues and portability problems were all documented and can be further reviewed in Chapter 5.

lyit | Institiúid Teicneolaíochta Leitir Ceannainn
Letterkenny Institute of Technology

Appendix H

Performance Measurements

H.1 Introduction

The following are a series of test results, which calculated the average connection times in milliseconds (ms) to Oracle when executing a particular query. Code Listing 5.1 in Chapter 5 indicates a piece of code that was embedded into the programs that would display the connection times to the screen. The tests were carried out over a ten-day period, at various times of the day and under various network loads.

Appendix F contains a series of Likert Scale questions, which were presented to 30 individuals who used the system and gave their responses to the questions accordingly. This was to test the viability of XML and applications of it. Chapter 5, Sections 5.4, 5.5, 5.6, 5.7 and 5.8 discuss the findings in greater detail. Table 5.2 in Chapter 5 also highlights some of the browser support issues that were raised when developing the systems. This was a problem that was envisaged from the beginning and was investigated as a result.

The MS Access connection was also tested by requested data of various data types to be retrieved from the database and displayed as XML. Chapter 5, Table 5.3 outlines what data types were returned successfully and which were not. A series of SQL statements were used in order to retrieve the information.

H.1.1 Testing the Applications

Tests 1 through to 19 demonstrate that XML can successfully interact with a legacy database system. In Chapter 1 Section 1.1.2, the 'Problem Definition' set out to establish whether or not XML could interoperate with a relational database. By using Java this was possible and connection times were relatively quick. However, please refer to Chapter 5 Section 5.9, for discussion on an anomaly that was discovered. Figure 5.4 depicts this anomaly occurring.

Test 1

Date: 4th June 2003

Time: 11:15am

Quantity: 50 files in the Database

	Specific Search	Select All	Add	Update
	110 ms	110 ms	140 ms	110 ms
	160 ms	100 ms	110 ms	110 ms
Average	135 ms	105 ms	125 ms	110 ms

Test 2

Date: 4th June 2003

Time: 6pm

Quantity: 50 files in the Database

	Specific Search	Select All	Add	Update
	110 ms	110 ms	110 ms	110 ms
	110 ms	110 ms	110 ms	110 ms
Average	110 ms	110 ms	110 ms	110 ms

Test 3Date: 5th June 2003

Time: 10:45am

Quantity: 50 files in the Database

	Specific Search	Select All	Add	Update
	201 ms	591 ms	110 ms	110 ms
	110 ms	110 ms	110 ms	110 ms
Average	155 ms	350 ms	110 ms	110 ms

Test 4Date: 5th June 2003

Time: 2:10pm

Quantity: 50 files in the Database

	Specific Search	Select All	Add	Update
	111 ms	120 ms	110 ms	110 ms
	130 ms	110 ms	110 ms	111 ms
Average	120 ms	115 ms	110 ms	110 ms

Test 5Date: 5th June 2003

Time: 4:15pm

Quantity: 50 files in the Database

	Specific Search	Select All	Add	Update
	110 ms	110 ms	110 ms	110 ms
	100 ms	110 ms	110 ms	111 ms
Average	105 ms	110 ms	110 ms	110 ms

Test 6Date: 6th June 2003

Time: 11:10am

Quantity: 50 files in the Database

	Specific Search	Select All	Add	Update
	110 ms	110 ms	120 ms	110 ms
	110 ms	100 ms	110 ms	110 ms
Average	110 ms	105 ms	115 ms	110 ms

Test 7Date: 6th June 2003

Time: 2pm

Quantity: 50 files in the Database

	Specific Search	Select All	Add	Update
	110 ms	110 ms	110 ms	110 ms
	110 ms	130 ms	111 ms	110 ms
Average	110 ms	120 ms	110 ms	110 ms

Test 8Date: 9th June 2003

Time: 11am

Quantity: 50 files in the Database

	Specific Search	Select All	Add	Update
	110 ms	111 ms	110 ms	110 ms
	110 ms	110 ms	110 ms	100 ms
Average	110 ms	110 ms	110 ms	105 ms

Test 9Date: 9th June 2003

Time: 2:45pm

Quantity: 50 files in the Database

	Specific Search	Select All	Add	Update
	120 ms	110 ms	110 ms	110 ms
	110 ms	100 ms	120 ms	111 ms
Average	115 ms	105 ms	115 ms	110 ms

Test 10Date: 10th June 2003

Time: 11am

Quantity: 50 files in the Database

	Specific Search	Select All	Add	Update
	111 ms	110 ms	100 ms	110 ms
	111 ms	110 ms	110 ms	110 ms
Average	111 ms	110 ms	105 ms	110 ms

Test 11Date: 10th June 2003

Time: 2pm

Quantity: 50 files in the Database

	Specific Search	Select All	Add	Update
	110 ms	110 ms	110 ms	110 ms
	110 ms	111 ms	110 ms	100 ms
Average	110 ms	110 ms	110 ms	105 ms

Test 12**Date: 10th June 2003****Time: 4:45pm****Quantity: 50 files in the Database**

	Specific Search	Select All	Add	Update
	101 ms	101 ms	100 ms	110 ms
	100 ms	131 ms	100 ms	111 ms
Average	100 ms	116 ms	100 ms	110 ms

Test 13**Date: 11th June 2003****Time: 10:45am****Quantity: 50 files in the Database**

	Specific Search	Select All	Add	Update
	110 ms	110 ms	110 ms	110 ms
	110 ms	110 ms	110 ms	100 ms
Average	110 ms	110 ms	110 ms	105 ms

Test 14**Date: 11th June 2003****Time: 2:30pm****Quantity: 50 files in the Database**

	Specific Search	Select All	Add	Update
	121 ms	110 ms	110 ms	110 ms
	110 ms	110 ms	110 ms	111 ms
Average	115 ms	110 ms	110 ms	110 ms

Test 15**Date: 12th June 2003****Time: 11am****Quantity: 50 files in the Database**

	Specific Search	Select All	Add	Update
	111 ms	110 ms	110 ms	110 ms
	110 ms	111 ms	110 ms	111 ms
Average	110 ms	110 ms	110 ms	110 ms

Test 16**Date: 12th June 2003****Time: 2pm****Quantity: 50 files in the Database**

	Specific Search	Select All	Add	Update
	110 ms	110 ms	110 ms	110 ms
	110 ms	110 ms	110 ms	110 ms
Average	110 ms	110 ms	110 ms	110 ms

Test 17**Date: 12th June 2003****Time: 4pm****Quantity: 50 files in the Database**

	Specific Search	Select All	Add	Update
	110 ms	110 ms	110 ms	110 ms
	121 ms	110 ms	110 ms	120 ms
Average	115 ms	110 ms	110 ms	115 ms

Test 18**Date: 16th June 2003****Time: 10:55am****Quantity: 50 files in the Database**

	Specific Search	Select All	Add	Update
	110 ms	110 ms	111 ms	110 ms
	110 ms	110 ms	120 ms	110 ms
	110 ms	111 ms	110 ms	110 ms
	110 ms	111 ms	121 ms	110 ms
	111 ms	110 ms	110 ms	111 ms
Average	110 ms	110 ms	115 ms	110 ms

Test 19**Date: 10th July 2003****Time: 10:30am****Quantity: 50 files in the Database**

	Specific Search	Select All	Add	Update
	120 ms	110 ms	110 ms	110 ms
	110 ms	110 ms	110 ms	111 ms
Average	115 ms	110 ms	110 ms	110 ms

Appendix I

Program Listing

lyit | Institiúid Teicneolaíochta Leitir Ceanaínn
Letterkenny Institute of Technology


```

*****
//Author: Nigel McKelvey
//Date: February 2003
//Referred to "Designing XML Databases", by Mark Graves
//for certain functions
//(refer to the References Chapter for further details)
*****
//First.java
package com.xweave.XMLDatabase;

// Contains defaults for system

public class First
{
    public static final int ORACLE = 1;

    //XMLAccount is the RDBMS account used for storing XML data
    //access string is jdbc:oracle:thin:acct/password@machine:port:instance for Oracle JDBC
    public static String XMLAccount =
    "jdbc:oracle:thin:NIGELMCKELVEY/nigel@172.18.1.27:1521:STUDENTC";

    //RelDatabaseAccount is the RDBMS account used
    //to retrieve additional relational data (if null, XMLAccount is used)
    public static String RelDatabaseAccount = null;

    public static String parserClass = "org.apache.xerces.parsers.SAXParser";
    public First()
    {
        // do not instantiate
    }
    public static int getDBProduct()
    {
        return DBProduct;
    }
    public static String getParserClass()
    {
        return parserClass;
    }
    public static String RelationalAccount()
    {
        if (RelDatabaseAccount == null)
        {
            return XMLAccount();
        }
        return RelDatabaseAccount;
    }
    public static String XMLAccount()
    {
        return XMLAccount;
    }
    public static void init()
    {
        //every main method in 'XMLDatabase' calls this method
        try
        {
            Integer integerValue = null;
            String val = null;
            integerValue = Integer.getInteger("xmldb.dbproduct", null);

```

```

        if (integerValue != null)
            setDBProduct(integerValue.intValue());
        val = System.getProperty("xmldb.saxparser", null);
        if (val != null)
            setParserClass(val);
        val = System.getProperty("xmldb.xmldbacct", null);
        if (val != null)
            setXmldbAcct(val);
        val = System.getProperty("xmldb.reldbacct", null);
        if (val != null)
            setRelldbAcct(val);
    }
    catch (Exception ex)
    {
        //catch exceptions, and ignore
    }
}
public static void setDBProduct(int newDBProduct)
{
    DBProduct = newDBProduct;
}
public static void setParserClass(String newValue)
{
    First.parserClass = newValue;
}
public static void setRelldbAcct(String newValue)
{
    First.RelDatabaseAccount = newValue;
}
public static void setXmldbAcct(String newValue)
{
    First.XMLAccount = newValue;
}
}

/*****

//Author: Nigel McKelvey
//Date: February 2003
//Referred to "Designing XML Databases", by Mark Graves
//for certain functions
//(refer to the References Chapter for further details)
// HandleDoc.java
package com.xweave.XMLDatabase.finegrainedRel;

import org.xml.sax.*;
import java.util.*;

// SAX Handler which creates SQL to load a document into the database

public class HandleDoc extends HandlerBase
{
    protected final static String DOC_TABLE = "XDB_DOC";
    protected final static String ELE_TABLE = "XDB_ELE";
    protected final static String ATTR_TABLE = "XDB_ATTR";
    protected final static String CHILD_TABLE = "XDB_CHILD";
    protected final static String STR_TABLE = "XDB_STR";
    protected final static String TEXT_TABLE = "XDB_TEXT";
    protected final static int CDATA_SPLIT_LENGTH = 255;
    private Element curEle = null;

```

```

private Stack elementStack = null;

// counters for element, attribute, and cdata may be generated
// within the class because they are only unique to the current
// document. The document identifier must come from the database.

protected int eleCtr = 1;
protected int attrCtr = 1;
protected int cdataCtr = 1;
protected String documentName = null;
protected String documentId = null;
public HandleDoc()
{
    super();
}
public HandleDoc(String documentId)
{
    super();
    setDocumentId(documentId);
}

// Handle character data regions.

public void characters(char[] chars, int start, int length)
{
    // First check to see if characters have whitespace whitespace,
    // if so, return
    boolean isWhitespace = true;
    for (int i = start; i < start+length; i++)
    {
        if (! Character.isWhitespace(chars[i]))
        {
            isWhitespace = false;
            break;
        }
    }
    if (isWhitespace == true)
    {
        return;
    }
    // Determine which CDATA table to use
    String table;
    boolean useStr = true;
    if (length <= CDATA_SPLIT_LENGTH)
    {
        table = STR_TABLE;
    }
    else
    {
        table = TEXT_TABLE;
        useStr = false;
    }
    // Create an 'insert' statement
    int currentId = nextCdataCtrValue();
    StringBuffer buf = new StringBuffer();
    buf.append("insert into");
    buf.append(" " + table + " ");
    buf.append("(doc_id, cdata_id, ele_id, val)");
    buf.append(" values ");
    buf.append("(" + getDocumentId() + ", " +

```

```

currentId + ", " +
getCurEle().getId() + ", ");

if (useStr)
{
    buf.append("");
    buf.append(chars, start, length);
    buf.append("");
}
else
{
    buf.append("");
    buf.append(chars, start, length);
    buf.append("");
}
buf.append("");
rdbExecute(buf.toString());
// create child entry for the new CDATA
buf = new StringBuffer();
buf.append("insert into");
buf.append(" " + CHILD_TABLE + " ");
buf.append("(doc_id, ele_id, indx, child_class, child_id)");
buf.append(" values ");
buf.append("(" + getDocumentId() + ", " +
            getCurEle().getId() + ", " +
            getCurEle().incrIndexCtr() + ", ");

if (useStr)
{
    buf.append("'STR', ");
}
else
{
    buf.append("'TEXT', ");
}
buf.append(currentId + ")");
rdbExecute(buf.toString());
}

public void endElement(String name)
{
    //clear 'curEle' and pop the stack
    if (getElementStack().empty())
    {
        this.curEle = null;
    }
    else
    {
        this.curEle = (Element) getElementStack().pop();
    }
}

public Element getCurEle()
{
    return curEle;
}

public String getDocumentId()
{
    return documentId;
}

public String getDocumentName()
{
    if (documentName == null)

```

```

        {
            setDocumentName("NoName" + getDocumentId());
        }
        return documentName;
    }
    private Stack getElementStack()
    {
        if (elementStack == null)
        {
            elementStack = new java.util.Stack();
        }
        return elementStack;
    }
    protected int nextAttrCtrValue()
    {
        return attrCtr++;
    }
    protected int nextCdataCtrValue()
    {
        return cdataCtr++;
    }
    protected int nextEleCtrValue()
    {
        return eleCtr++;
    }

    //All SQL output goes through the method below:
    public boolean rdbExecute(String val)
    {
        // add a semicolon to print to standard out
        System.out.println(val + ";");
        return true;
    }
    protected void setCurrentElement(String newValue)
    {
        if (getCurEle() != null)
        {
            //push the parent element onto the stack
            getElementStack().push(getCurEle());
        }
        this.curEle = new Element(newValue);
    }
    public void setDocumentId(String newValue)
    {
        this.documentId = newValue;
    }
    public void setDocumentName(String newValue)
    {
        this.documentName = newValue;
    }
    private void setElementStack(Stack newValue)
    {
        this.elementStack = newValue;
    }

    // Initialise the document table when starting an XML document

    public void startDocument()
    {

```

```

//create a document entry
StringBuffer buf = new StringBuffer();
buf.append("insert into");
buf.append(" " + DOC_TABLE + " ");
buf.append("(doc_id, name)");
buf.append(" values ");
buf.append("(" + getDocumentId() + ", '" + getDocumentName() + "'");
rdbExecute(buf.toString());
}

// Handle element, attributes, and the connection from this element
// to its parent

public void startElement(String name, AttributeList attrList)
{
    //Create new element entry
    Element parent = getCurEle();
    String parentId;
    if (parent == null)
    {
        parentId = "NULL";
    }
    else
    {
        parentId = parent.getId();
    }
    setCurrentElement(Integer.toString(nextEleCtrValue()));
    String currentId = getCurEle().getId();
    StringBuffer buf = new StringBuffer();
    buf.append("insert into");
    buf.append(" " + ELE_TABLE + " ");
    buf.append("(doc_id, ele_id, parent_id, tag)");
    buf.append(" values ");
    buf.append("(" + getDocumentId() + ", " +
                currentId + ", " +
                parentId + ", " +
                "'" + name + "'");

    rdbExecute(buf.toString());
    if (parent == null)
    {
        //if it is a root element, update the document entry
        buf = new StringBuffer();
        buf.append("update");
        buf.append(" " + DOC_TABLE + " ");
        buf.append("set root = " + currentId);
        buf.append(" where doc_id = " + getDocumentId());
        rdbExecute(buf.toString());
    }
    else
    {
        //if it is not a root element, then create a child entry
        buf = new StringBuffer();
        buf.append("insert into");
        buf.append(" " + CHILD_TABLE + " ");
        buf.append("(doc_id, ele_id, indx, child_class, child_id)");
        buf.append(" values ");
        buf.append("(" + getDocumentId() + ", " +
                parentId + ", " +
                parent.incrIndexCtr() + ", " +
                "ELE', " +

```

```

                                currentId + "));
        rdbExecute(buf.toString());
    }
    //Create entries for every attribute
    for (int i = 0; i < attrList.getLength(); i++)
    {
        buf = new StringBuffer();
        buf.append("insert into");
        buf.append(" " + ATTR_TABLE + " ");
        buf.append("(doc_id, attr_id, ele_id, name, val)");
        buf.append(" values ");
        buf.append("(" + getDocumentId() + ", " +
                    nextAttrCtrValue() + ", " +
                    currentId + ", " +
                    "" + attrList.getName(i) + ", " +
                    "" + attrList.getValue(i) + ")");

        rdbExecute(buf.toString());
    }
}
}
}

```

```

/*****

```

```

//Author: Nigel McKelvey
//Date: February 2003
//Referred to "Designing XML Databases", by Mark Graves
//for certain functions (refer to the References Chapter for further details)
// LoadXML.java
package com.xweave.XMLDatabase.finegrainedRel;

import com.xweave.XMLDatabase.util.rdb.RelationalDatabase;
import com.xweave.XMLDatabase.util.io.Output;
import java.sql.*;
import org.xml.sax.*;
import org.xml.sax.helpers.ParserFactory;
import org.w3c.dom.Document;
import java.io.IOException;

```

```

// Command to Load the XML document into the database

```

```

public class LoadXML extends Command
{
    public LoadXML()
    {
        super();
    }
    public LoadXML(RelationalDatabase rdb)
    {
        super(rdb);
    }
    public LoadXML(RelationalDatabase rdb, Output output)
    {
        super(rdb,output);
    }
    public HandleDoc newHandler()
    {
        if (getRdb() == null)
        {

```



```

        return new HandleDoc();
    }
    else
    {
        return new JDBCLoadHandler(getRdb());
    }
}

//Parse an XML file using a SAX parser
//the parser in <parserClass> should be placed on the Java CLASSPATH

public String parse(String xmlFile)
{
    return parse(xmlFile, null);
}

public String parse(String xmlFile, String name)
{
    String parserClass = com.xweave.XMLDatabase.First.getParserClass();
    try
    {
        Parser parser = ParserFactory.makeParser(parserClass);
        HandlerBase handler = this.newHandler();
        parser.setDocumentHandler(handler);
        parser.setErrorHandler(handler);
        try
        {
            //determine whether the file is a document or a URL
            String prefix = xmlFile.substring(0,7);
            if (prefix.startsWith("http://") ||
                prefix.startsWith("file://") ||
                prefix.startsWith("ftp://"))
            {
                //URL
                if (name == null)
                {
                    ((HandleDoc)
                    handler).setDocumentName(xmlFile);
                }
                else
                {
                    ((HandleDoc)
                    handler).setDocumentName(name);
                }
                parser.parse(xmlFile);
            }
            else
            {
                //File
                if (name != null)
                {
                    ((HandleDoc)
                    handler).setDocumentName(name);
                }
                parser.parse(new InputSource(new
                java.io.StringReader(xmlFile)));
            }
            return ((HandleDoc) handler).getDocumentId();
        }
    }
    catch (SAXException se)

```

```

        {
            getOutput().writeln(se.toString());
            se.printStackTrace();
        }
        catch (IOException ioe)
        {
            ioe.printStackTrace();
        }
    }
    catch (ClassNotFoundException ex)
    {
        ex.printStackTrace();
    }
    catch (IllegalAccessException ex)
    {
        ex.printStackTrace();
    }
    catch (InstantiationException ex)
    {
        ex.printStackTrace();
    }
    return null;
}

// Parse an XML file using a SAX parser
public String parseText(String xmlText, String name)
{
    String parserClass = com.xweave.XMLDatabase.First.getParserClass();
    try
    {
        Parser parser = ParserFactory.makeParser(parserClass);
        HandlerBase handler = this.newHandler();
        parser.setDocumentHandler(handler);
        parser.setErrorHandler(handler);
        try
        {
            ((HandleDoc) handler).setDocumentName(name);
            parser.parse(new InputSource(new
                java.io.StringReader(xmlText)));
            return ((HandleDoc) handler).getDocumentId();
        }
        catch (SAXException se)
        {
            getOutput().writeln(se.toString());
            se.printStackTrace();
        }
        catch (IOException ioe)
        {
            ioe.printStackTrace();
        }
    }
    catch (ClassNotFoundException ex)
    {
        ex.printStackTrace();
    }
    catch (IllegalAccessException ex)
    {
        ex.printStackTrace();
    }
}

```

```

        catch (InstantiationException ex)
        {
            ex.printStackTrace();
        }
        return null;
    }

    public String parseUrl(String xmlUrl, String name)
    {
        String parserClass = com.xweave.XMLDatabase.First.getParserClass();
        try
        {
            Parser parser = ParserFactory.makeParser(parserClass);
            HandlerBase handler = this.newHandler();
            parser.setDocumentHandler(handler);
            parser.setErrorHandler(handler);
            try
            {
                ((HandleDoc) handler).setDocumentName(name);
                parser.parse(xmlUrl);
                return ((HandleDoc) handler).getDocumentId();
            }
            catch (SAXException se)
            {
                getOutput().writeln(se.toString());
                se.printStackTrace();
            }
            catch (IOException ioe)
            {
                ioe.printStackTrace();
            }
        }
        catch (ClassNotFoundException ex)
        {
            ex.printStackTrace();
        }
        catch (IllegalAccessException ex)
        {
            ex.printStackTrace();
        }
        catch (InstantiationException ex)
        {
            ex.printStackTrace();
        }
        return null;
    }
}

```

```

//*****

```

```

//Author: Nigel McKelvey

```

```

//Date: February 2003

```

```

//Referred to "Designing XML Databases", by Mark Graves

```

```

//for certain functions (refer to the References Chapter for further details)

```

```

// RunApp.java

```

```

package com.xweave.XMLDatabase.finegrainedRel;

```

```

import com.xweave.XMLDatabase.util.rdb.*;

```

```

import com.xweave.XMLDatabase.util.io.*;

```

```

import java.awt.*;

```

```

import java.lang.*;
import java.util.*;
import java.io.*;

//Demonstrate a Fine Grained relational storage

public class RunApp
{
    protected LoadXML load = null;
    protected Alter format = null;
    private RelationalDatabase rdb = null;
    private ListDocs documentList = null;
    protected String AccessRDB = com.xweave.XMLDatabase.First.XMLAccount();
    protected final static String DOC_TABLE = "XDB_DOC";
    protected final static String USAGE =
    "demo (help |list| connect <connectstring> | store <url> | retrieve <id>)*";
    protected Output output = null;
    public RunApp()
    {
        super();
    }
    public void connect()
    {
        try
        {
            setRdb(new RDBConnector(new JDBCACct(AccessRDB)));
        }
        catch (java.sql.SQLException ex)
        {
            ex.printStackTrace();
        }
    }
    public void connect(String val)
    {
        this.setRdbAccessString(val);
        connect();
    }

    public boolean dispatchArg(String command, String val)
    {
        if (command.equalsIgnoreCase("RETRIEVE"))
        {
            this.retrieveDocumentId(val);
            return true;
        }
        if (command.equalsIgnoreCase("RETRIEVEDOC"))
        {
            this.getOutput().writeln("<?xml version='1.0'?>");
            this.retrieveDocumentId(val);
            return true;
        }
        if (command.equalsIgnoreCase("STORE"))
        {
            this.storeDocument(val);
            return true;
        }
        if (command.equalsIgnoreCase("CONNECT"))
        {
            this.connect(val);
            return true;
        }
    }
}

```

```

    }
    if (command.equalsIgnoreCase("HELP"))
    {
        this.getOutput().writeln(USAGE);
        return true;
    }
    return false;
}

// Get the Load command object

public ListDocs getDocumentList()
{
    if(documentList == null)
    {
        documentList = new ListDocs(getRdb(), getOutput());
    }
    return documentList;
}

public LoadXML getLoad()
{
    if (load == null)
    {
        if (getRdb() == null)
        {
            load = new LoadXML();
        }
        else
        {
            load = new LoadXML(getRdb());
        }
    }
    return load;
}

public Alter getFormat()
{
    if(format == null)
    {
        format = new Alter(getRdb(),getOutput());
    }
    return format;
}

public Output getOutput()
{
    if (output == null)
    {
        setOutput(new Output());
    }
    return output;
}

public RelationalDatabase getRdb()
{
    if (rdb == null)
    {

```

```

        if (getRdbAccessString() != null)
        {
            //create a connection to Oracle
            connect();
        }
    }
    return rdb;
}
public String getRdbAccessString()
{
    return AccessRDB;
}

//Starts the application.
//Takes database commands as arguments

public static void main(java.lang.String[] args)
{
    if (args.length == 0)
    {
        System.err.println(USAGE);
    }
    int ctr = 0;
    String command;
    com.xweave.XMLDatabase.First.init();
    RunApp demo = new RunApp();
    boolean connected = false;
    while (ctr < args.length)
    {
        command = args[ctr++];
        if (command.equalsIgnoreCase("HELP"))
        {
            System.err.println(USAGE);
            continue;
        }
        if (command.equalsIgnoreCase("LIST"))
        {
            demo.writeListDocs();
            continue;
        }
        if (command.equalsIgnoreCase("CONNECT"))
        {
            demo.connect(args[ctr++]);
            connected = true;
            continue;
        }
        if (command.equalsIgnoreCase("STORE"))
        {
            demo.storeDocument(args[ctr++]);
            continue;
        }
        if (command.equalsIgnoreCase("RETRIEVE"))
        {
            demo.retrieveDocumentId(args[ctr++]);
            continue;
        }
    }
    if (connected)
    {

```

```

        try
        {
            RelationalDatabase rdb = demo.rdb;
            if (rdb != null)
            {
                rdb.close();
            }
        }
        catch (Throwable e)
        {
            e.printStackTrace();
        }
    }

    // Alter the XML document with id <documentId>

    public boolean retrieveDocumentId(String documentId)
    {
        return getFormat().writeDoc(documentId);
    }
    public void setOutput(Output newValue)
    {
        this.output = newValue;
    }
    protected void setRdb(RelationalDatabase newValue)
    {
        this.rdb = newValue;
    }
    public void setRdbAccessString(String newValue)
    {
        this.AccessRDB = newValue;
    }

    // Print a status message

    public void status(String msg)
    {
        System.err.println(msg);
    }

    //Store a document from the text <xmlDocText>
    //include valid XML processing instructions

    public String storeDocText(String xmlDocText, String name)
    {
        String documentId = null;
        try
        {
            documentId = getLoad().parseText(xmlDocText, name);
        }
        catch (Throwable ex)
        {
            status("Error: " + ex.getMessage());
        }
        if (documentId != null)
        {
            status("Loaded document " + documentId + " from text");
        }
        else
    }

```



```

        {
            status("Could not load document from text");
        }
        return documentId;
    }

    // Store a document from the given URL <xmlFile>
    public String storeDocument(String xmlFile)
    {
        return storeDocument(xmlFile, null);
    }

    // Store a document from the given URL <xmlFile>
    public String storeDocument(String xmlFile, String name)
    {
        String documentId = null;
        java.util.Date dbase_startTime = new java.util.Date();

        try
        {
            documentId = getLoad().parse(xmlFile, name);
        }
        catch (Throwable ex)
        {
            status("Error: " + ex.getMessage());
        }
        if (documentId != null)
        {
            status("Loaded document " + documentId + " from " + xmlFile);
        }
        else
        {
            status("Could not load document from " + xmlFile);
        }
        return documentId;
    }

    // Store a document from the given URL <xmlFile>
    public String storeDocUrl(String xmlFile, String name)
    {
        String documentId = null;

        try
        {
            documentId = getLoad().parseUrl(xmlFile, name);
        }
        catch (Throwable ex)
        {
            status("Error: " + ex.getMessage());
        }

        if (documentId != null)
        {
            status("Loaded document " + documentId + " from " + xmlFile);
        }
        else
    }

```

```

        {
            status("Could not load document from " + xmlFile);
        }
        return documentId;
    }

    // Write a list (to out) of all XML documents

    public void writeListDocs() {
        getDocumentList().writeListDocs();
    }
}

```

```

/*****

```

```

// Output.java

```

```

package com.xweave.XMLDatabase.util.io;

```

```

import java.io.*;

```

```

// Handles output to an output source

```

```

public class Output

```

```

{
    public Output()
    {
        super();
    }
    public void write(String s)
    {
        System.out.print(s);
    }
    public void writeln(String s)
    {
        System.out.println(s);
    }
}

```

```

/*****

```

```

// OracleDB.java

```

```

package com.xweave.XMLDatabase.util.rdb;

```

```

import java.sql.*;

```

```

// Allows access to Oracle

```

```

// Requires JDBC Thin Driver to be in the class path.

```

```

public class OracleDB extends RDBConnector implements RelationalDatabase

```

```

{
    public OracleDB() throws SQLException
    {
        super();
        setConnectionStringPrefix("jdbc:oracle:thin:");
    }
    public OracleDB(RDBAcct acct) throws SQLException
    {
        super();
        SetAccount(acct);
        setConnectionStringPrefix("jdbc:oracle:thin:");
        connect();
    }
}

```

```

}

//*****
// RelationalDatabase.java
package com.xweave.XMLDatabase.util.rdb;

import java.sql.*;

// Interface to access a Relational Database

public interface RelationalDatabase
{
    void close();
    public boolean connect(Connection conn);
    public boolean connect(RDBAcct acct);
    public boolean connect(Connection conn);
    ResultSet executeQuery(String stmt) throws SQLException;
    int executeUpdate(String stmt) throws SQLException;
    String getDataItem(String query);
    ResultSet getData(String query);
    String getDataItem(String query);
    public boolean isClosed();
}

//*****
// RDBAcct.java
package com.xweave.XMLDatabase.util.rdb;

// Account and access for relational database.

public interface RDBAcct
{
    String getAcct();
    void SetAccount(String acct);
    String toString();
}

//*****
//ListDocs.java
package com.xweave.XMLDatabase.finegrainedRel;

import com.xweave.XMLDatabase.util.io.*;
import com.xweave.XMLDatabase.util.rdb.*;

//List the document in the database

public class ListDocs extends Command
{
    // ListDocs constructor comment.

    public ListDocs()
    {
        super();
    }

    public ListDocs(RelationalDatabase rdb)
    {
        super(rdb);
    }
}

```

```

    }

    public ListDocs(RelationalDatabase rdb, Output out)
    {
        super(rdb, out);
    }

    // Return a list of all the XML documents

    public String getListDocs()
    {
        StringBuffer buf = new StringBuffer();
        java.util.Enumeration e = retrieveDocumentListVector().elements();
        while (e.hasMoreElements())
        {
            buf.append((String) e.nextElement());
            buf.append("\n");
        }
        return buf.toString();
    }
    // Return a Vector of all the XML documents with their ids
    // as a Vector of Strings with format <id> <space> <name>

    public java.util.Vector retrieveDocumentListVector()
    {
        java.util.Vector vec = new java.util.Vector();
        StringBuffer buf = new StringBuffer();
        buf.append("select doc_id, name from ");
        buf.append(DOC_TABLE);
        buf.append(" order by doc_id");
        try
        {
            java.sql.ResultSet rset = getRdb().getData(buf.toString());
            while (rset.next())
            {
                // for each document, append <id> <space> <name>
                vec.addElement(rset.getString(1) + " " + rset.getString(2));
            }
            return vec;
        }
        catch (java.sql.SQLException ex)
        {
            ex.printStackTrace();
            return null;
        }
    }

    //Write a list of all the XML documents

    public void writeListDocs()
    {
        java.util.Enumeration e = retrieveDocumentListVector().elements();
        Output out = getOutput();
        while (e.hasMoreElements())
        {
            out.writeln((String) e.nextElement());
        }
    }

```

```

// Write a list of all the XML documents (as select options)

public void writeDocumentSelect()
{
    java.util.Enumeration e = retrieveDocumentListVector().elements();
    Output out = getOutput();
    while (e.hasMoreElements())
    {
        out.writeln("<option>" + (String) e.nextElement() + "</option>");
    }
}

}

//*****
// Command.java
package com.xweave. XMLDatabase.finegrainedRel;

import com.xweave. XMLDatabase.util.io.*;
import com.xweave. XMLDatabase.util.rdb.*;
import java.sql.SQLException;

public abstract class Command
{
    protected final static String DOC_TABLE = "XDB_DOC";
    protected final static String ELE_TABLE = "XDB_ELE";
    protected final static String ATTR_TABLE = "XDB_ATTR";
    protected final static String CHILD_TABLE = "XDB_CHILD";
    protected final static String TEXT_TABLE = "XDB_TEXT";
    protected final static String STR_TABLE = "XDB_STR";
    private Output output = null;
    private RelationalDatabase rdb = null;

    public Command()
    {
        super();
    }

    public Command(RelationalDatabase rdb)
    {
        super();
        setRdb(rdb);
    }

    public Command(RelationalDatabase rdb, Output out)
    {
        super();
        setRdb(rdb);
        setOutput(out);
    }

    public Output getOutput()
    {
        return output;
    }

    protected RelationalDatabase getRdb()
    {
        return rdb;
    }

    public void rdbExecute(String val) throws SQLException

```

```

    {
        // add a semicolon to print to standard out, omit for JDBC
        if (getRdb() == null)
        {
            System.out.println(val + ";");
        }
        else
        {
            getRdb().executeUpdate(val);
        }
    }
    public void setOutput(Output newValue)
    {
        this.output = newValue;
    }
    protected void setRdb(RelationalDatabase newValue)
    {
        this.rdb = newValue;
    }
    public void write(String val)
    {
        getOutput().write(val);
    }
    public void writeln(String val)
    {
        getOutput().writeln(val);
    }
}

```

```

//*****

```

```

// Alter.java

```

```

package com.xweave. XMLDatabase.finegrainedRel;

```

```

import com.xweave. XMLDatabase.util.rdb.RDB;
import com.xweave. XMLDatabase.util.io.Output;
import java.util.*;
import java.sql.*;

```

```

public class Alter extends Command

```

```

{
    protected final static int CDATA_SPLIT_LENGTH = 255;
    protected String document;
    protected boolean writeElementId = false;

    public Alter()
    {
        super();
    }
    public Alter(RelationalDatabase rdb)
    {
        super(rdb);
    }
    public Alter(RelationalDatabase rdb, Output output)
    {
        super(rdb, output);
    }
    public boolean getWriteElementId()
    {
        return writeElementId;
    }
}

```

```

}
public void setWriteElementId(boolean newValue)
{
    this.writeElementId = newValue;
}
public void write(String val, int dep)
{
    write(val);
}
protected void writeCdata(String elementId, String cdataId, String table, int dep) throws
SQLException
{
    StringBuffer buf = new StringBuffer();
    buf.append("select val from");
    buf.append(" " + table + " ");
    buf.append("where doc_id = " + document);
    buf.append(" and ele_id = " + elementId);
    buf.append(" and cdata_id = " + cdataId);
    String val = getRdb().getDataItem(buf.toString());
    writeln(val, dep);
}

// Send an element child

protected void writeChild(String childClass, String elementId, String childId, int childDepth)
throws SQLException
{
    if (childClass.equalsIgnoreCase("ELE"))
    {
        //call writeElement recursively
        writeElement(childId, childDepth);
    }
    else if (childClass.equalsIgnoreCase("STR"))
    {
        //writeCdata works for STR or TEXT using JDBC
        writeCdata(elementId, childId, STR_TABLE, childDepth);
    }
    else if (childClass.equalsIgnoreCase("TEXT"))
    {
        //writeCdata works for STR or TEXT using JDBC
        writeCdata(elementId, childId, TEXT_TABLE, childDepth);
    }
    else
    {
        //default in case other classes are added to the database
        writeln("<unknown class=\"" + childClass + "\"
id=\"" + childId + "\"/>", childDepth);
    }
}

// Retrieve a document from the database and write it

public boolean writeDoc(String document)
{
    this.document = document;
    StringBuffer buf = new StringBuffer();
    buf.append("select root from");
    buf.append(" " + DOC_TABLE + " ");

```



```

        buf.append(" where doc_id = " + document);
    try
    {
        String elementId = getRdb().getDataItem(buf.toString());
        writeElement(elementId, 0);
    }
    catch (SQLException ex)
    {
        ex.printStackTrace();
        return false;
    }
    return true;
}

// Write an element from the database and its children

protected void writeElement(String elementId, int dep) throws SQLException
{
    StringBuffer buf = new StringBuffer();
    buf.append("select tag from");
    buf.append(" " + ELE_TABLE + " ");
    buf.append("where doc_id = " + document);
    buf.append(" and ele_id = " + elementId);
    String tagName = getRdb().getDataItem(buf.toString());
    //write start tag
    write("<" + tagName, dep);
    //write element id
    if (getWriteElementId())
    {
        write(" FRAGID=\" + document + "." + elementId + "\"");
    }
    //write attributes
    buf = new StringBuffer();
    buf.append("select name, val from");
    buf.append(" " + ATTR_TABLE + " ");
    buf.append("where doc_id = " + document);
    buf.append(" and ele_id = " + elementId);
    ResultSet rset = getRdb().getData(buf.toString());
    if (rset != null) {
        while (rset.next())
        {
            // for each attribute, write name and val
            write(" " + rset.getString(1) + "=\"" + rset.getString(2) + "\"");
        }
        rset.close();
    }
    //close start tag, and check to see if there are any children
    buf = new StringBuffer();
    buf.append("select indx, child_class, child_id from");
    buf.append(" " + CHILD_TABLE + " ");
    buf.append("where doc_id = " + document);
    buf.append(" and ele_id = " + elementId);
    rset = getRdb().getData(buf.toString());
    boolean moreRows = rset != null && rset.next();
    if (moreRows)
    {
        // has children, close start tag and continue
        writeln(">");
    }
    else

```

```

    {
        // has no children, close tag and return
        writeln(">");
        return;
    }
    // write children
    // retrieve all rows, then recurse down
    // to prevent leaving cursor open while recursing further
    Vector vec = new Vector();
    int index;
    while (moreRows)
    {
        //rows are ordered by index
        index = Integer.parseInt(rset.getString(1));
        if (index > vec.size())
        {
            vec.setSize(index);
        }
        vec.insertElementAt(new Child(rset.getString(2), rset.getString(3)), index);
        moreRows = rset.next();
    }
    if (rset != null) rset.close();
    //write each child
    Enumeration e = vec.elements();
    Child child;
    String childClass;
    int childDepth = dep + 1;
    while (e.hasMoreElements())
    {
        child = (Child) e.nextElement();
        if (child == null)
        {
            continue;
        }
        childClass = child.getClassName();
        writeChild(childClass, elementId, child.getId(), childDepth);
    }
    //write end tag
    writeln("<" + tagName + ">");
}

// Retrieve a single element from the database and write it
public boolean writeFragment(String document, String element)
{
    this.document = document;
    try
    {
        writeElement(element, 0);
    }
    catch (SQLException ex)
    {
        ex.printStackTrace();
        return false;
    }
    return true;
}
public void writeln(String val, int dep)
{
    writeln(val);
}

```

lyit | Institiúid Teicneolaíochta Leitir Ceanaínn
Letterkenny Institute of Technology

```

    }
}

//*****
// RDBConnector.java
package com.xweave.XMLDatabase.util.rdb;

import java.sql.*;

// Allows access to a relational database
// Requires JDBC Thin Driver in the class path.

// Drivers are registered from the array 'Drivers'.

public class RDBConnector implements RelationalDatabase
{
    protected Statement[] stmt = new Statement[32];
    protected RDBAcct acct = null;
    protected Connection conn = null;

    //connectStringPrefix = "jdbc:oracle:thin:"

    public String connectStringPrefix = "";

    public String connectStringSuffix = "";
    public String[] Drivers =
    {
        "oracle.jdbc.driver.OracleDriver",
        "COM.ibm.db2.jdbc.app.DB2Driver"
    };
    public RDBConnector() throws SQLException
    {
        super();
        register();
    }
    public RDBConnector(RDBAcct acct) throws SQLException
    {
        super();
        SetAccount(acct);
        register();
        connect();
    }
    public void close()
    {
        try
        {
            if (conn.isClosed())
            {
                return;
            }
            else
            {
                conn.close();
            }
        }
        catch (SQLException ex)
        {
            ex.printStackTrace();
        }
    }
}

```

```

public boolean connect()
{
    try
    {
        if (conn != null)
        {
            return true;
        }
        conn = DriverManager.getConnection(getConnectionString());
        return true;
    }
    catch (SQLException ex)
    {
        ex.printStackTrace();
        return false;
    }
}

public boolean connect(RDBAcct acct)
{
    SetAccount(acct);
    return connect();
}

public boolean connect(Connection newConnection)
{
    conn = newConnection;
    return true;
}

public ResultSet executeQuery(String query) throws SQLException
{
    return executeQuery(query,0);
}

public ResultSet executeQuery(String query, int statementNum) throws SQLException
{
    if (stmt[statementNum] == null)
    {
        if (conn == null)
        {
            System.err.println("Database is closed for " + query);
            return null;
        }
        stmt[statementNum] = conn.createStatement();
    }
    return stmt[statementNum].executeQuery(query.trim());
}

// execute a SQL statement that does not return a result
// return the number of rows changed, or 0 if there is no result

public int executeUpdate(String sql) throws SQLException
{
    try
    {
        if (stmt[0] == null)
        {
            if (conn == null)
            {
                System.err.println("Database is closed for " + sql);
                return 0;
            }
            stmt[0] = conn.createStatement();
        }
    }
}

```

```

        }
        return stmt[0].executeUpdate(sql.trim());
    }
    catch (SQLException ex)
    {
        ex.printStackTrace();
        return 0;
    }
}
public RDBAcct getAcct()
{
    return acct;
}
public java.sql.Connection getConnection()
{
    //Provide direct access to RelationalDatabase Connection.
    return conn;
}
public String getConnectionString()
{
    return getConnectionStringPrefix() + getAcct() + getConnectionStringSuffix();
}
public String getConnectionStringPrefix()
{
    return connectionStringPrefix;
}
public String getConnectionStringSuffix()
{
    return connectionStringSuffix;
}
public ResultSet getData(String query)
{
    return getData(query,0);
}
public ResultSet getData(String query, int statementNum)
{
    try
    {
        return executeQuery(query, statementNum);
    }
    catch (SQLException ex)
    {
        ex.printStackTrace();
        return null;
    }
    catch (Exception ex)
    {
        ex.printStackTrace();
        return null;
    }
}

// Get the singleton value of a query

public String getDataItem(String query)
{
    //returns the value of the first column of the first row
    try
    {
        ResultSet resultSet = getData(query);

```

```

        if (resultSet == null)
        {
            //error in query

            return null;
        }
        resultSet.next();
        int type = resultSet.getMetaData().getColumnType(1);
        String result = resultSet.getString(1);
        resultSet.close();
        if (type == java.sql.Types.LONGVARCHAR)
        {
            //Correct a problem with Oracle JDBC handling of LONGs

            getStatement().close();
            stmt[0] = null;
        }
        return result;
    }
    catch (SQLException ex)
    {
        ex.printStackTrace();
        return null;
    }
}

public Statement getStatement()
{
    return stmt[0];
}

public Statement getStatement(int statementNum)
{
    return stmt[statementNum];
}

public boolean isClosed()
{
    if (conn == null)
    {
        return true;
    }
    else
    {
        try
        {
            return conn.isClosed();
        }
        catch (SQLException ex)
        {
            ex.printStackTrace();
            return true;
        }
    }
}

public void register()
{
    //registers all the drivers in Drivers
    for (int i = 0; i < Drivers.length; i++)
    {
        try
        {
            Class.forName(Drivers[i]);
        }
    }
}

```



```

        }
        catch (Exception ex)
        {
            //ignore failed drivers
        }
    }
}

public void SetAccount(JDBCAcct newValue)
{
    this.acct = newValue;
}

public void SetAccount(RDBAcct newValue)
{
    this.acct = newValue;
}

public void setConnectionStringPrefix(String newValue)
{
    this.connectionStringPrefix = newValue;
}

public void setConnectionStringSuffix(String newValue)
{
    this.connectionStringSuffix = newValue;
}
}

}

//*****
// WorkRDB.java
package com.xweave.XMLDatabase.util.rdb;

import com.xweave.XMLDatabase.util.rdb.*;
import com.xweave.XMLDatabase.util.io.*;

// Interacts with RDB database and Output IO.

public abstract class WorkRDB
{
    private RelationalDatabase rdb = null;
    protected String AccessRDB = com.xweave.XMLDatabase.First.XMLAccount();
    protected Output output = null;
    public WorkRDB()
    {
        super();
    }
    public void connect()
    {
        try
        {
            setRdb(new RDBConnector(new JDBCAcct(AccessRDB)));
        }
        catch (java.sql.SQLException ex)
        {
            ex.printStackTrace();
        }
    }
    public void connect(String val)
    {
        this.setRdbAccessString(val);
        connect();
    }
    public Output getOutput()

```

```

    {
        if (output == null)
        {
            setOutput(new Output());
        }
        return output;
    }
    public RelationalDatabase getRdb()
    {
        if (rdb == null)
        {
            if (getRdbAccessString() != null)
            {
                //create a connection to RelationalDatabase
                connect();
            }
        }
        return rdb;
    }
    public String getRdbAccessString()
    {
        return AccessRDB;
    }
    public void setOutput(Output newValue)
    {
        this.output = newValue;
    }
    protected void setRdb(RelationalDatabase newValue)
    {
        this.rdb = newValue;
    }
    public void setRdbAccessString(String newValue)
    {
        this.AccessRDB = newValue;
    }
}

//*****
rem LoadURL.bat
rem This is a batch file which is used in order to load an
rem XML file into the Oracle database.

SET XMLDB_HOME=.
SET XMLDB_JAR="C:\DBMasters\java\xml db.jar"
SET XMLSAX_JAR="C:\DBMasters\java\xerces.jar"
SET JDBC_JAR="C:\DBMasters\oracle_jdbc_80520"
SET CONNECT="jdbc:oracle:thin:NIGELMCKELVEY/nigel@172.18.1.27:1521:STUDENTC"
java -cp "%XMLDB_JAR%;%XMLSAX_JAR%;%JDBC_JAR%" com.xweave.
XMLDatabase.finegrainedRel.RunApp connect "%CONNECT%" store
file:///C:/DBMasters/ch4/A30.xml

//*****

```

```
// JDBC-ODBC Connection

/*****
/* Author: Nigel McKelvey
/* Date: January 2003
/* Description: This program acts as a 'gateway'
/* to the data in the database. It calls the
/* ToXML function.
/* Reference:
/* --- "Professional XML Databases" by
/* --- Kevin Williams.
*****/

// Gateway.java
package com.resources.jdbc;

//Libraries

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class Gateway extends HttpServlet
{
    //Allow the XMLDataGateway to respond to HTTP
    //GET requests

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
    IOException, ServletException
    {
        //set the MIME type in order to
        //establish that we are working with XML

        response.setContentType("text/xml");

        //Need to be able to write a response
        //back to the client

        PrintWriter out = response.getWriter();

        //instantiate the JDBC2XML class

        ToXML searchObj = new ToXML();

        //get the desired information

        out.println(searchObj.execute(request.getParameter("driver"),
        request.getParameter("jdbcurl"), request.getParameter("uid"),
        request.getParameter("pwd"), request.getParameter("sql")));
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response) throws
    IOException, ServletException
    {
        doGet(request, response);
    }
}

```

```

/*****
/*Author: Nigel McKelvey */
/*Date: January 2003 */
/*Description: This program helps to retrieve */
/* data from an Access database and display */
/* the result in XML format. The information */
/* is instantly given meaning and thus made */
/* more useful on the Internet. */
*****/

// ToXML.java
package com.resources.jdbc;

import java.sql.*;
import java.awt.*;
import java.io.*;
import java.util.*;

public class ToXML
{
    public ToXML()
    {
        super();
    }

    //Bring everything together by executing the
    //search and return the result

    public String execute(String driver, String url, String uid, String pwd, String sql)
    {
        String output = new String();

        try
        {
            //instantiate and register the JDBC driver
            Class.forName(driver);
            //connect to the database and create a statement
            Connection conn = DriverManager.getConnection(url, uid, pwd);
            Statement s = conn.createStatement();

            //execute the SQL statement

            ResultSet ResSet = s.executeQuery(sql);
            output = WriteTheXML(ResSet);

            //close the connections

            ResSet.close();
            conn.close();

            //catch any exceptions and return the error message

        }
        catch(Exception e)
        {
            output = "<error>" + encodeXML(e.toString()) + "</error>";
        }
        return output;
    }
}

```

```

}
//A single quote in a SQL statement is doubled in order
//to keep the system from crashing

public String SQLEncode(String content)
{
    return Replace(content, "\"", "\"\"");
}
//XML encoding rules to special characters which are
//similar to those in HTML
//Do a 'search and replace' of the characters

String encodeXML(String sData)
{
    String[] before = {"&", "<", ">", "\"", "\""};
    String[] after = {"&amp;", "&lt;", "&gt;", "&quot;", "&apos;"};
    if(sData!=null) {
        for(int z=0; z<before.length; z++)
        {
            sData = Replace(sData, before[z], after[z]);
        }
    }
    return sData;
}
//Replace all occurrences of oldWord with newWord in content

String Replace(String content, String oldWord, String newWord)
{
    int position = content.indexOf(oldWord);

    //Loop through the string 'content'

    while (position > -1)
    {
        content = content.substring(0,position) + newWord +
        content.substring(position+oldWord.length());
        position = content.indexOf(oldWord,position+newWord.length());
    }
    return content;
}

//Serialise the JDBC result set to XML

String WriteTheXML(ResultSet ResSet)
{
    //Create the XML code
    //Need a StringBuffer object to hold the output

    StringBuffer strResults = new StringBuffer("<?xml version=\"1.0\" encoding=\"ISO-8859-1\"?>\r\n<resultset>\r\n");
    try
    {
        ResultSetMetaData rsMetadata = ResSet.getMetaData();
        int Fields = rsMetadata.getColumnCount();
        strResults.append("<metadata>\r\n");
    }
}

```

```

//create a field for each column in the DB

for(int y=1; y <= Fields; y++)
{
    strResults.append("<field name=\"" + rsMetadata.getColumnname(h) +
        "\" datatype=\"" + rsMetadata.getColumnTypeName(h) + "\"/>\r\n");
}
strResults.append("</metadata>\r\n<records>\r\n");

//create a record element

while(ResultSet.next())
{
    strResults.append("<record>\r\n");
    for(int x=1; x <= Fields; x++)
    {
        strResults.append("<field name=\"" +
            rsMetadata.getColumnname(i) + "\">" +
            encodeXML(ResultSet.getString(i)) + "</field>\r\n");
    }
    strResults.append("</record>\r\n");
}
}
catch(Exception e) {}
strResults.append("</records>\r\n</resultset>");
return strResults.toString();
}
}

```

```

/*****
/* Author: Nigel McKelvey
/* Date: January 2003
/* Description: This program helps to retrieve
/* data from an Access database and display
/* the result in XML format. The information
/* is instantly given meaning and thus made
/* more useful on the Internet. The main
/* difference between this program and the
/* ToXML.java is that this program uses
/* the WebRowSet package which gives the data
/* additional metadata.
*****/

```

```

// RowSetServlet.java
package com.resources.jdbc;

```

```

//Libraries

```

```

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import javax.sql.*;
import sun.jdbc.rowset.*;
import java.sql.*;

```

```

public class RowSetServlet extends HttpServlet
{

```

```

public void doGet(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException
{
    //set the MIME type to deal with XML

    response.setContentType("text/xml");
    PrintWriter out = response.getWriter();
    try
    {
        //instantiate and register the JDBC driver

        Class.forName(request.getParameter("driver"));
        WebRowSet Webrs;

        // create a new row set
        Webrs = new WebRowSet();

        // set the properties of the rowset

        Webrs.setUrl(request.getParameter("jdbcurl"));
        Webrs.setUsername(request.getParameter("uid"));
        Webrs.setPassword(request.getParameter("pwd"));
        Webrs.setCommand(request.getParameter("sql"));

        Webrs.setTransactionIsolation(java.sql.Connection.TRANSACTION_READ_UNCOMMITTED);

        // populate the rowset

        Webrs.execute();
        Webrs.WriteTheXML(out);
    }
    catch(Exception e)
    {
        throw new ServletException(e);
    }
}

public void doPost(HttpServletRequest request, HttpServletResponse response) throws
IOException, ServletException
{
    doGet(request, response);
}
}

//*****
<!--index.html -->
<!-- The HTML Page that provides links to the JDBC-ODBC Application -->

<html>
<head>
<title>By Nigel McKelvey</title>
</head>

<body bgcolor="beige">
<H2 align="center"><font face="Arial">Connecting to MS Access through JDBC and returning the
result in XML</H2>

<br><br>
<p><A href="xmlsqlquery.html">

```


XML SQL Query

: This example allows you to query a remote JDBC data source. The results are returned as XML.

</p>

<p>

WebRowSet XML SQL Query

:

This example also allows you to query a remote JDBC data source. The results are returned as XML. This particular example uses the WebRowSet class.

</p>

<p><center><h4>SMIL Presentation</h4></center>

<p align="center">

<object id="media"

classid="CLSID:CFCDAA03-8BE4-11CF-B84B-0020AFBCCFA" width="550" height="400">

<param name="src" value="X:/menu/main_FOR_TOMCAT2.smi">

<param name="console" value="Clip1">

<param name="controls" value="ImageWindow">

<param name="AutoStart" value="TRUE">

<embed controls="ImageWindow" console="Clip1" type="audio/x-pn-realaudio-plugin" src="X:/menu/main_FOR_TOMCAT2.smi" width="250" height="188" autostart="true">

</embed>

</object>

<center>

<EMBED SRC="X:/menu/HomeLink.svg" NAME="SVGEmbed" HEIGHT="33" WIDTH="97" TYPE="image/svg+xml" PLUGINSOURCE="http://www.adobe.com/svg/viewer/install/">

<FORM ACTION="bogus_submit.html" METHOD="POST">

<INPUT NAME="data" TYPE="hidden">

</FORM>

</center>

</body>

</html>

// main_FOR_TOMCAT2.smi

//The SMIL file that presents information to users

//on how to connect to the database

<smil xmlns="http://www.w3.org/2001/SMIL20/Language">

<head>

<meta name="title" content="SMIL Presentation" />

<meta name="author" content="CBT Application" />

<meta name="copyright" content="(c) Nigel McKelvey 2002" />

<!-- position the media elements on the screen -->

<layout>

<root-layout width="500" height="400" backgroundColor="black" />

```

<region id="flash" width="400" height="200" left="50" top="100" z-index="1" />
<region id="rp" width="196" height="152" left="25" top="50" z-index="4" />
<region id="vid" width="196" height="152" left="150" top="125" z-index="3" />
<region id="img" width="196" height="152" left="275" top="200" z-index="2" />
<region id="logo" width="170" height="81" left="300" top="40" z-index="5" />
<region id="text" width="300" height="175" left="25" top="225" z-index="1" />
<regPoint id="centered" regAlign="center" left="50%" top="50%" />
</layout>

<!-- slide transition effect -->

<transition id="upSlide" type="slideWipe" subtype="fromBottom" />
<transition id="ellipse" type="ellipseWipe" subtype="horizontal" />
<transition id="rightSlide" type="slideWipe" subtype="fromRight" />
<transition id="leftSlide" type="slideWipe" subtype="fromLeft" />
<transition id="fadeBrown" type="fade" subtype="fadeToColor" fadeColor="#CC9966"/>
<transition id="fadeOrange" type="fade" subtype="fadeToColor" fadeColor="#FFCC66"/>
<transition id="fadeBeige" type="fade" subtype="fadeToColor" fadeColor="#FFFFCC"/>

</head>

<body>

<par dur="indefinite">

<!-- position text from a real text file at the left hand side of the screen -->

<brush id="rp1" color="#FFFFCC" transIn="leftSlide" begin="00:05.5" region="rp"
fill="hold" />
<brush id="vid1" color="#FFCC66" transIn="leftSlide" begin="00:06.5" region="vid"
fill="hold" />
<brush id="img1" color="#CC9966" transIn="leftSlide" begin="00:07.5" region="img"
fill="hold" />

<textstream src="text2.rt" begin="00:08.0" region="text" fill="hold" />

<excl dur="indefinite">

<par begin="00:09.0" >

<!-- position the icons on the slides -->


<param name="bitrate" value="1765"/>
</img>

<param name="bitrate" value="1765"/>
</img>

<param name="bitrate" value="1765" />
</img>
</par>

```

```

<par id="yellow" begin="rp1.activateEvent">
  <set targetElement="rp" attributeName="z-index" to="4"/>
  <set targetElement="vid" attributeName="z-index" to="2"/>
  <set targetElement="img" attributeName="z-index" to="3"/>

  <!-- play the audio file in conjunction with the 'slides' -->

  <audio src="audio/access_left.wav" />
  <seq end="vid1.activateEvent; rp1.activateEvent">
    
      <param name="bitrate" value="1765"/>
    </img>
    
      <param name="bitrate" value="1765"/>
    </img>
    
      <param name="bitrate" value="1765"/>
    </img>
    
      <param name="bitrate" value="1765"/>
    </img>
    
      <param name="bitrate" value="1765"/>
    </img>
    
      <param name="bitrate" value="1765"/>
    </img>
    
      <param name="bitrate" value="1765"/>
    </img>
    
      <param name="bitrate" value="1765"/>
    </img>
    <brush color="#FFFFCC" region="rp" fill="hold"/>
    
      <param name="bitrate" value="1765" />
  </img>
  </seq>
</par>

<par id="orange" begin="vid1.activateEvent">
  <set targetElement="vid" attributeName="z-index" to="4"/>
  <set targetElement="rp" attributeName="z-index" to="2"/>
  <set targetElement="img" attributeName="z-index" to="3"/>

  <!-- play the video taking into consideration the bitrate -->

  <video src="video/nigel02.avi" region="vid" transIn="ellipse"
    transOut="fadeOrange" regPoint="centered" fill="remove" />
  <brush color="#CC9966" region="img" fill="hold"/>

```

```


  <param name="bitrate" value="1765" />
</img>

<brush color="#FFFFCC" region="rp" fill="hold"/>

</img>
</par>

<!-- play the audio file in conjunction with the 'slides' -->

<par id="brown" begin="img1.activateEvent">
  <set targetElement="rp" attributeName="z-index" to="2"/>
  <set targetElement="vid" attributeName="z-index" to="3"/>
  <set targetElement="img" attributeName="z-index" to="4"/>
  <audio src="audio/access_right.wav" />
  <seq end="vid1.activateEvent; rp1.activateEvent">
  
  <param name="bitrate" value="1765"/>
  </img>
  
  <param name="bitrate" value="1765"/>
  </img>

  
  <param name="bitrate" value="1765"/>
  </img>
  
  <param name="bitrate" value="1765"/>
  </img>
  
  <param name="bitrate" value="1765"/>
  </img>
  
  <param name="bitrate" value="1765"/>
  </img>
  <brush color="#CC9966" region="img" fill="hold"/>
  
  <param name="bitrate" value="1765" />
  </img>
  <brush color="#CC9966" region="img" fill="hold"/>
  
  <param name="bitrate" value="1765" />
  </img>
  <brush color="#FFCC66" region="vid" fill="hold"/>
  
  <param name="bitrate" value="1765" />
  </img>

```

```

        <brush color="#FFFFCC" region="rp" fill="hold"/>
        
          <param name="bitrate" value="1765" />
        </img>
      </seq>
    </par>
  </excl>
</par>
</body>
</smil>

```

```

//*****

```

```

<!--xmlsqlquery.html -->

```

```

<html>

```

```

<head>

```

```

<meta http-equiv="Content-Type"

```

```

  content="text/html; charset=iso-8859-1">

```

```

<title>XML Gateway to the JDBC Data Source</title>

```

```

</head>

```

```

<body bgcolor="beige">

```

```

<p align="center"><font size="4" face="Arial"><strong>XML Gateway to the JDBC Data
Source</strong></font></p>

```

```

<p><font face="Arial"></font>&nbsp;</p>

```

```

<form action="/jdbcxml/servlet/Gateway" method="POST">

```

```

  <table border="0">

```

```

    <!--Set up the text fields -->

```

```

      <tr>

```

```

        <td align="right"><font face="Arial">JDBC Driver: </font></td>

```

```

        <td><font face="Arial">

```

```

          <input type="text" size="55" name="driver"

```

```

          value="sun.jdbc.odbc.JdbcOdbcDriver"></td>

```

```

        </tr>

```

```

      <tr>

```

```

        <td align="right"><font face="Arial">JDBC URL: </font></td>

```

```

        <td><font face="Arial">

```

```

          <input type="text" size="55" name="jdbcurl" value="jdbc:odbc:Nigel"></td>

```

```

        </tr>

```

```

      <tr>

```

```

        <td align="right"><font face="Arial">Userid:</font></td>

```

```

        <td><font face="Arial"><input type="text" size="55"

```

```

          name="uid"></font></td>

```

```

      </tr>

```

```

      <tr>

```

```

        <td align="right"><font face="Arial">password </font></td>

```

```

        <td><font face="Arial"><input type="password" size="55"

```

```

          name="pwd"></font></td>

```

```

      </tr>

```

```

      <tr>

```

```

        <td align="right"><font face="Arial">SQL Statement:</font></td>

```

```

        <td><textarea name="sql" rows="10" cols="55"></textarea></td>
    </tr>
    <tr>
        <td align="right"><input type="submit"></td>
        <td>&nbsp;</td>
    </tr>
</table>

<!-- Link to Menu -->

<br><p><A href="X:\menu\openingScreen3.html">Home Page</A>
<p><A href="index.html">Main Database Menu</A>
</form>

</body>
</html>

//*****
<!-- webrowset.html -->
<html XMLNS:t="urn:schemas-microsoft-com:time">

<head>

<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1">
<title>WebRowSet JDBC Search</title>
</head>

<body bgcolor="beige">

<p align="center"><font size="5" face="Arial"><strong>WebRowSet JDBC
Search</strong></font></p>

<p><font face="Arial">&nbsp;</font></p>

<form action="/jdbcxml/servlet/RowSetServlet" method="POST">
    <table border="0">
        <tr>
            <td align="right"><font face="Arial">JDBC Driver: </font></td>
            <td><font face="Arial">
                <input type="text" size="50" name="driver"
                value="sun.jdbc.odbc.JdbcOdbcDriver">
            </td>
        </tr>
        <tr>
            <td align="right"><font face="Arial">JDBC URL: </font></td>
            <td><font face="Arial"> *
                <input type="text" size="50" name="jdbcurl" value="jdbc:odbc:Nigel">
            </td>
        </tr>
        <tr>
            <td align="right"><font face="Arial">Userid:</font></td>
            <td><font face="Arial"><input type="text" size="50"
            name="uid"></font></td>
        </tr>
        <tr>
            <td align="right"><font face="Arial">password </font></td>
            <td><font face="Arial"><input type="password"

```

```
        size="50" name="pwd"></font></td>
</tr>

<tr>
    <td align="right"><font face="Arial">SQL Statement:</font></td>
    <td><textarea name="sql" rows="10" cols="50"></textarea></td>
</tr>

<tr>
    <td align="right"><input type="submit"></td>
    <td>&nbsp;</td>
</tr>

</table>
<br><p><a href="X:\menu\openingScreen3.html">Home Page</a>
<p><a href="index.html">Main Database Menu</a>
</form>
</body>
</html>
```

```
//*****
```

lyit | Institiúid Teicneolaíochta Leitir Ceanaínn
Letterkenny Institute of Technology


```
//Applications of XML
```

```
/**
*****
** SVG **
*/
```

```
<!-- HomeLink.svg -->
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN"
"http://www.w3.org/TR/2000/CR-SVG-20001102/DTD/svg-20001102.dtd">

<svg width="100%" height="100%"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://www.w3.org/2000/svg">
  <rect x="0" width="100" height="33" style="fill:beige;"/>

  <a xlink:href="x:\Menu\openingScreen3.html" id="HomeLink">
    <text style="fill:blue; stroke:black; font-size:16;" x="9" y="19">
      Home Page
    </text>
  </a>

</svg>
```

```
/**
*****
** HTML PAGE WITH SVG EMBEDDED WITHIN IT **
*/
```

```
<!-- Opening3.html -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>By Nigel McKelvey 2002</TITLE>
</HEAD>
<BODY BGCOLOR="beige">
<center>
```

```
<!-- Call the SVG Page -->
```

```
<EMBED SRC="eg3.svg" NAME="SVGEmbed" HEIGHT="390" WIDTH="390" TYPE="image/svg-
xml" PLUGINSPPAGE="http://www.adobe.com/svg/viewer/install/">

<FORM ACTION="bogus_submit.html" METHOD="POST">
  <INPUT NAME="data" TYPE="hidden">
</FORM>
</center>
</BODY>
</HTML>
```

```
/**
*****
** HTML PAGE WITH SVG EMBEDDED WITHIN IT **
*/
```

```
<!-- openingScreen2.html -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>By Nigel McKelvey 2002</TITLE>
</HEAD>
<BODY BGCOLOR="beige">
<center>
```

```
<!-- Call the SVG file -->
```

```
<EMBED SRC="eg2.svg" NAME="SVGEmbed" HEIGHT="400" WIDTH="500" TYPE="image/svg-
xml" PLUGINSPPAGE="http://www.adobe.com/svg/viewer/install/">
```

```

<FORM ACTION="bogus_submit.html" METHOD="POST">
  <INPUT NAME="data" TYPE="hidden">
</FORM>
</center>
</BODY>
</HTML>

//*****
/** SVG **/
<!-- eg2.svg -->
<?xml version="1.0" encoding="iso-8859-1" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20001102//EN"
  "http://www.w3.org/TR/2000/CR-SVG-20001102/DTD/svg-20001102.dtd" >

<!-- ***** -->
<!-- Nigel McKelvey -->
<!-- 2003 -->
<!-- This SVG Program presents animation to the -->
<!-- user with a 'falling stars' effect! -->
<!-- ***** -->

<svg width="500px" height="350px">

  <!-- to make the background color="beige" -->

  <rect width="100%" height="100%" style="fill:#555555"/>
  <rect width="500px" height="350px" style="fill:#555555"/>
  <defs>

    <!-- svg object: star -->

    <g id="star" transform="scale(2)">
      <polygon points="0,-14 8.229,11.326 -13.315,-4.326 13.315,-4.326 -8.229,11.326"/>
    </g>

  </defs>

  <!-- add text: XML Technologies In Use!!! -->

  <text x="90" y="90" style="fill:white; font-size:30pt"> XML </text>
  <text x="240" y="140" style="fill:white; font-size:30pt"> Technologies </text>
  <text x="320" y="240" style="fill:white; font-size:30pt"> In </text>
  <text x="300" y="340" style="fill:white; font-size:30pt"> Use ! </text>
  <text x="240" y="270" text-anchor="end" class="fil9 fnte hidden" style="fill:#45FE21; font-size:20pt;
  stroke:BLACK">
    <a xlink:href="x:\menu\openingScreen3_2.html">Enter Home Page</a>
    <animate attributeName="visibility" attributeType="CSS" to="1" begin="10s" dur="0s"
    fill="freeze" />
  </text>

  <!-- svg object: gold star -->
  <!-- specify the size, colour etc of the stars -->

  <use xlink:href="#star" x="150" y="160" style="fill:yellow;fill-rule:nonzero;stroke:black;">

```

```

<animateMotion dur="3s" repeatCount="indefinite" path="M 0,-360 0,0"/>
</use>
<use xlink:href="#star" x="150" y="160" style="fill:yellow;fill-rule:nonzero;stroke:black;">
  <animateMotion dur="3s" repeatCount="indefinite" path="M 350,0 0,0"/>
</use>
<use xlink:href="#star" x="150" y="160" style="fill:yellow;fill-rule:nonzero;stroke:black;">
  <animateMotion dur="3s" repeatCount="indefinite" path="M -300,0 0,0"/>
</use>
<use xlink:href="#star" x="150" y="160" style="fill:gold;fill-rule:nonzero;stroke:black;"/>
<use xlink:href="#star" x="150" y="160" style="fill:yellow;fill-rule:nonzero;stroke:black;">
  <animateMotion dur="3s" repeatCount="indefinite" path="M 0,0 350,0"/>
</use>
<use xlink:href="#star" x="150" y="160" style="fill:yellow;fill-rule:nonzero;stroke:black;">
  <animateMotion dur="3s" repeatCount="indefinite" path="M 0,0 -300,0"/>
</use>
<use xlink:href="#star" x="150" y="160" style="fill:yellow;fill-rule:nonzero;stroke:black;">
  <animateMotion dur="3s" repeatCount="indefinite" path="M 0,0 0,-360"/>
</use>
</svg>

/*****
** SVG **
<!-- eg3.svg -->
<?xml version="1.0" encoding="iso-8859-1"?>

<!-- ***** -->
<!-- Nigel McKelvey -->
<!-- 2003 -->
<!-- This SVG Program will count down from -->
<!-- 10 to 0 in a similar way to Flash. -->
<!-- the user can then click on the link to enter -->
<!-- ***** -->

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000303 Stylable//EN"
"http://www.w3.org/TR/2000/03/WD-SVG-20000303/DTD/svg-20000303-stylable.dtd" [

  <!-- Set some attribute values -->

  <!ENTITY st0 "opacity:0.86;">
  <!ENTITY st1 "font-size:135;">
  <!ENTITY st2 "opacity:0.77;fill:#D9E6BF;stroke:BLACK;">
  <!ENTITY st3 "opacity:0.43;stroke:GREEN;">
  <!ENTITY st4 "fill-rule:nonzero;clip-rule:nonzero;stroke:GREEN;stroke-miterlimit:4;">
  <!ENTITY st5 "stroke:GREEN; text-anchor:middle;">
  <!ENTITY st6 "font-family:'Comic Sans';">
]>
<svg width="200pt" height="200pt" viewBox="0 0 200 200" xmlns:a="http://www.adobe.com/svg10-
extensions"
  a:timeline="independent">
  <style type="text/css">

  </style>
  <g id="Layer_x0020_1" style="&st4; &st6; &st1;">

<rect width="100%" height="100%" style="&st2;">

```

```
<animate begin="5.5s" attributeName="fill" from="#D9E6BF" to="yellow" dur="0.1s"
repeatCount="0"/></rect>
```

```
<!-- Set the path for the graphics to be made available within -->
```

```
<path style="&st3;" d="M50,43c-0.009,24.873,1.166,50.25,2.75"/>
<path style="&st3;"
d="M106,40c3.924,4.052,4.848,10.884,7.178,16.203c4.575,10.441,11.959,19.562,17.145,29.646c6.22
7,12.107,12.104,24.589,19.924,35.952C160.521,136.733,169.98,149.332,177,166"/>
<path style="&st3;" d="M36,181c9.82-3.662,17.638-12.784,27.223-
17.274C73.374,158.971,83.011,152.219,93,148"/>
<path style="&st3;" d="M80,33C66.531,21.19,48.145,15.928,33,7"/>
<path style="&st3;" d="M129,26c15.436-6.071,31.349-14.97,46-23"/>
<path style="&st3;" d="M194,84c-27.997,10.579-55.758,23.422-85,30"/>
<path style="&st3;" d="M13,83c10.932,8.798,20.537,19.808,32,28"/>
```

```
<!-- Display the numbers, specifying their position -->
```

```
<!-- and their time on the screen -->
```

```
<g id="textLayout1" transform="translate(100, 150)" style="&st5;" >
<text style="opacity:0;">
  <animate begin="0.5s" attributeName="opacity" from="0.86" to="0"
dur="0.5s" repeatCount="0"/>10
</text>
<text style="opacity:0;">
  <animate begin="1s" attributeName="opacity" from="0.86" to="0"
dur="0.5s" repeatCount="0"/>9
</text>
<text style="opacity:0;">
  <animate begin="1.5s" attributeName="opacity" from="0.86" to="0"
dur="0.5s" repeatCount="0"/>8
</text>
<text style="opacity:0;">
  <animate begin="2s" attributeName="opacity" from="0.86" to="0"
dur="0.5s" repeatCount="0"/>7
</text>
<text style="opacity:0;">
  <animate begin="2.5s" attributeName="opacity" from="0.86" to="0"
dur="0.5s" repeatCount="0"/>6
</text>
<text style="opacity:0;">
  <animate begin="3s" attributeName="opacity" from="0.86" to="0"
dur="0.5s" repeatCount="0"/>5
</text>
<text style="opacity:0;">
  <animate begin="3.5s" attributeName="opacity" from="0.86" to="0"
dur="0.5s" repeatCount="0"/>4
</text>
<text style="opacity:0;">
  <animate begin="4s" attributeName="opacity" from="0.86" to="0"
dur="0.5s" repeatCount="0"/>3
</text>
<text style="opacity:0;">
  <animate begin="4.5s" attributeName="opacity" from="0.86" to="0"
dur="0.5s" repeatCount="0"/>2
</text>
<text style="opacity:0;">
  <animate begin="5s" attributeName="opacity" from="0.86" to="0"
dur="0.5s" repeatCount="0"/>1
```

```

        </text>
    </g>
</g>

    <!-- Create the link -->

    <text x="170" y="170" text-anchor="end" class="fil9 fnte hidden" style="fill:#45FE21; font-size:14pt; stroke:BLACK">
    <a xlink:href="openingScreen2.html">Enter Web Page</a>
    <animate attributeName="visibility" attributeType="CSS" to="1" begin="10s" dur="0s" fill="freeze" />

</text>
</svg>

/**
** HTML PAGE WITH SVG EMBEDDED WITHIN IT **
<!-- openingScreen3.html -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>By Nigel McKelvey 2002 - Home Page</TITLE>
</HEAD>
<BODY BGCOLOR="beige">
<center>

<!-- Call the SVG File -->

<EMBED SRC="intro.svg" NAME="SVGEmbed" HEIGHT="430" WIDTH="710"
TYPE="image/svg+xml" PLUGINSPPAGE="http://www.adobe.com/svg/viewer/install/">

<FORM ACTION="bogus_submit.html" METHOD="POST">
    <INPUT NAME="data" TYPE="hidden">
</FORM>
</center>
</BODY>
</HTML>

/**
** SVG - WITH GRAPHICS, ANIMATION, LINKS, ETC. **
<!-- intro.svg -->
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg onload="SetTime(evt)" width="100%" height="100%">

<!-- ***** -->
<!-- Author: Nigel McKelvey -->
<!-- Date: 2002 -->
<!-- This is the main interface to the -->
<!-- system. It contains graphics, animation and -->
<!-- links created in SVG. -->
<!-- ***** -->

<title>Nigel McKelvey 2002</title>
<style type="text/css">
    .rbtn{fill:#D9E6BF;stroke:#000000;stroke-width:1;filter:url(#Bevel);}
    .tbtn{fill:#000000;font-size:12;font-family:Arial;text-anchor:middle;}

```

```

</style>
<defs>
<!-- Java Script to get the current time form the machine -->

<script language="Javascript">
  <![CDATA[

function SetTime(LoadEvent)
{
  var Now = new Date();

  var Seconds = Now.getSeconds();
  var Minutes = Now.getMinutes() + Seconds / 60;
  var Hours = Now.getHours() + Minutes / 60;

  var SVGDocument = LoadEvent.getTarget().getOwnerDocument();

  SVGDocument.getElementById("seconds").setAttribute('transform', 'rotate(' + (Seconds * 6) +
  ');');
  SVGDocument.getElementById("minutes").setAttribute('transform', 'rotate(' + (Minutes * 6) +
  ');');
  SVGDocument.getElementById("hours").setAttribute('transform', 'rotate(' + (Hours * 30) + ')');
}

  ]>
</script>

  <!-- ball used in the animation -->

  <symbol id="ball">
    <circle cx="300" cy="20" r="20" fill="cyan" stroke="black" style="stroke-width:
    1;" />
  </symbol>

  <!-- Design the gradient for the background -->

  <linearGradient id="design" x1="0%" y1="0%" x2="100%" y2="0%"
    spreadMethod="pad" gradientUnits="objectBoundingBox">

    <!-- use the opacity attribute for transparency -->

    <stop offset="0%" style="stop-color:#555555;stop-opacity:0.6"/>
    <stop offset="40%" style="stop-color:#555565;stop-opacity:1"/>
    <stop offset="90%" style="stop-color:#534555;stop-opacity:0.9"/>
    <stop offset="98%" style="stop-color:#D9E6BF;stop-opacity:0.8"/>
    <stop offset="100%" style="stop-color:#c0c0ff;stop-opacity:1"/>
    <stop offset="999%" style="stop-color:#000000;stop-opacity:0.5"/>
  </linearGradient>

  <filter id="Bevel" filterUnits="objectBoundingBox" x="-10%" y="-10%"
    width="150%" height="150%">
    <feGaussianBlur in="SourceAlpha" stdDeviation="3" result="blur"/>
    <feSpecularLighting in="blur" surfaceScale="3" specularConstant="0.5"
    specularExponent="5" result="specOut"
    style="lighting-color:rgb(255,245,250)">
    <fePointLight x="-7000" y="-20000" z="15000"/>
  </feSpecularLighting>
  <feComposite in="specOut" in2="SourceAlpha" operator="in" result="specOut2"/>
  <feComposite in="SourceGraphic" in2="specOut2" operator="arithmetic" k1="0"

```



```

        k2="1" k3="1" k4="0" result="litPaint"/>
    </filter>
    <!-- svg object: star -->

    <g id="star" transform="scale(2)">
        <polygon points="0,-14 8.229,11.326 -13.315,-4.326 13.315,-4.326 -8.229,11.326"/>
    </g>

</defs>

<!-- Apply the gradient colour to the background -->

<rect x="0" y="0" width="100%" height="100%"
      style="fill:url(#design);stroke:#000000;stroke-width:1"/>
</g>

<!-- Create the clock text -->

<circle cx="80" cy="80" r="65" style="fill:green;stroke:black;stroke-width:3"/>
<text x="104" y="36" style="font-size:12;fill:purple">1</text>
<text x="125" y="55" style="font-size:12;fill:purple">2</text>
<text x="135" y="86" style="font-size:12;fill:purple">3</text>
<text x="128" y="113" style="font-size:12;fill:purple">4</text>
<text x="110" y="131" style="font-size:12;fill:purple">5</text>
<text x="78" y="143" style="font-size:12;fill:purple">6</text>
<text x="47" y="133" style="font-size:12;fill:purple">7</text>
<text x="27" y="114" style="font-size:12;fill:purple">8</text>
<text x="17" y="86" style="font-size:12;fill:purple">9</text>
<text x="25" y="57" style="font-size:12;fill:purple">10</text>
<text x="45" y="35" style="font-size:12;fill:purple">11</text>
<text x="72" y="27" style="font-size:12;fill:purple">12</text>
<g transform="translate(80 80)">

<g id="hours">
    <line x1="0" y1="0" x2="0" y2="-35" style="stroke-width:4;stroke:black">
        <animateTransform attributeName="transform" type="rotate" dur="43200s"
            values="0;360" repeatCount="indefinite"/>
    </line>
</g>
<g id="minutes">
<line x1="0" y1="0" x2="0" y2="-55" style="stroke-width:2;stroke:black">

<!-- move the hands on the clock -->

<animateTransform attributeName="transform" type="rotate" dur="3600s" values="0;360"
    repeatCount="indefinite"/>
</line>
</g>
<g id="seconds">
<line x1="0" y1="0" x2="0" y2="-55" style="stroke-width:1;stroke:red">
    <animateTransform attributeName="transform" type="rotate" dur="60s" values="0;360"
        repeatCount="indefinite"/>
</line>
</g>
</g>
<circle cx="80" cy="80" r="3" style="fill:black;stroke:black"/>

```



```

<!-- svg object: gold star -->

<use xlink:href="#star" x="420" y="280" style="fill:beige;fill-rule:nonzero;">
  <animateMotion dur="3s" repeatCount="indefinite" path="M 0,-360 0,0"/>
</use>
<use xlink:href="#star" x="420" y="280" style="fill:lightgrey;fill-rule:nonzero;">
  <animateMotion dur="3s" repeatCount="indefinite" path="M 350,0 0,0"/>
</use>
<use xlink:href="#star" x="420" y="280" style="fill:beige;fill-rule:nonzero;">
  <animateMotion dur="3s" repeatCount="indefinite" path="M -300,0 0,0"/>
</use>
<use xlink:href="#star" x="420" y="280" style="fill:grey;fill-rule:nonzero;"/>
<use xlink:href="#star" x="420" y="280" style="fill:lightgrey;fill-rule:nonzero;">
  <animateMotion dur="3s" repeatCount="indefinite" path="M 0,0 350,0"/>
</use>
<use xlink:href="#star" x="420" y="280" style="fill:lightgrey;fill-rule:nonzero;">
  <animateMotion dur="3s" repeatCount="indefinite" path="M 0,0 -300,0"/>
</use>
<use xlink:href="#star" x="420" y="280" style="fill:beige;fill-rule:nonzero;">
  <animateMotion dur="3s" repeatCount="indefinite" path="M 0,0 0,-360"/>
</use>

<!-- Headings and Links that appear on the page -->

<text x="300px" y="40px" style="font-family:Arial Black;fill:gold;font-size:30;stroke:red">Home Page </text>

<use xlink:href="#ball" transform="translate(180, 100)" id="play">
  <animate attributeName="y" begin="click" dur="2s" values="0; -80; 0; 10; 0; -7; 0; 3; 0; -2; 0; 1; 0" />
</use>
<text x="195px" y="100" style="fill:gold;font-size:16;font-family:Arial">Click on the ball
and see it bounce !</text>

<a xlink:href="x:\menu\HELP_PAGES_VML.html" style="fill:gold">
  <text x="580px" y="100" style="fill:gold;font-size:22;font-family:Arial;stroke:red">Help !</text>
</a>

<a xlink:href="x:\menu\make tutorial\Alternative.html" style="fill:gold">
  <text x="500px" y="210" style="fill:gold;font-size:16;font-family:Arial">Alternative Tutorials !</text>
</a>

<!-- Main Links -->

<a xlink:href="http://localhost:8080/jdbcxml/index.html" style="fill:#eacc65">
  <text x="60px" y="170px" style="fill:gold;font-size:16;font-family:Arial">Look at
the MS Access Database</text>
</a>

<text x="300px" y="170px" style="fill:gold;font-size:16;font-family:Arial">or</text>

<a xlink:href="http://127.0.0.1:8080/jdbcxml/index.html" style="fill:#eacc65">
  <text x="320px" y="170px" style="fill:gold;font-size:16;font-family:Arial">from
here</text>

```

```

</a>

<a xlink:href="x:\menu\play time\play time.html" style="fill:#eae665">
  <text x="500px" y="170px" style="fill:gold;font-size:16;font-family:Arial">Simple
  Entertainment !</text>
</a>

<a xlink:href="NewSlideShow.html" style="fill:#eae665">
  <text x="60px" y="210px" style="fill:gold;font-size:16;font-family:Arial">Slide
  Show Presentation on SVG</text>
</a>

<a xlink:href="AllTutorials\tutorialxml.xml" style="fill:#eae665">
  <text x="500px" y="250px" style="fill:gold;font-size:16;font-family:Arial">Tutorial
  Summary</text>
</a>

<a xlink:href="Make Tutorial\EnlargePhotos.xml" style="fill:#eae665">
  <text x="60px" y="250px" style="fill:gold;font-size:16;font-
  family:Arial">XML and Photos!</text>
</a>

<a xlink:href="Applet4Search.html" style="fill:#eae665">
  <text x="60px" y="290px" style="fill:gold;font-size:16;font-family:Arial">Search
  through an XML File</text>
</a>

<a xlink:href="index.htm" style="fill:#eae665">
  <text x="60px" y="330px" style="fill:gold;font-size:16;font-family:Arial">View a
  Data Island</text>
</a>

  <text x="200px" y="330px" style="fill:gold;font-size:16;font-
  family:Arial">or</text>

<a xlink:href="x:\Menu\Personal_Database\Default.htm" style="fill:#eae665">
  <text x="225px" y="330px" style="fill:gold;font-size:16;font-
  family:Arial">Implementing XSL/CSS</text>
</a>

<a xlink:href="x-cart/x-cart.html" style="fill:#eae665">
  <text x="60px" y="370px" style="fill:gold;font-size:16;font-family:Arial">Possible
  XML Implementation</text>
</a>

<a xlink:href="book.html" style="fill:#eae665">
  <text x="60px" y="410px" style="fill:gold;font-size:16;font-
  family:Arial">Temporary XML Database</text>
</a>

<svg width="320" height="28" x="420" y="310">

<!-- Create the buttons -->

<a id="btn1" xlink:href="Coll.html">

```

```

    <rect class="rbtn" x="0" y="0" width="80" height="28">
    <set begin="btn1.mouseover" end="btn1.mouseout" attributeName="fill" to="#F2BC68" />
    </rect>
    <text class="tbtn" x="40" y="20px">SVG Page</text>
</a>

<a id="btn2" xlink:href="SimpleAnimation.svg">
    <rect class="rbtn" x="160" y="0" width="80" height="28">
    <set begin="btn2.mouseover" end="btn2.mouseout" attributeName="fill" to="#F2BC68" />
    </rect>
    <text class="tbtn" x="200" y="20px">Samples</text>
</a>

<a id="btn3" xlink:href="smilpresentation.html">
    <rect class="rbtn" x="160" y="0" width="80" height="28">
    <set begin="btn3.mouseover" end="btn3.mouseout" attributeName="fill" to="#F2BC68" />
    </rect>
    <text class="tbtn" x="200" y="20px">SMIL</text>
</a>
</svg>

```

```

<!-- Create the email envelope and perform animation -->

```

```

<svg x="500" y="370" width="80" height="45">

```

```

<!-- Email Image specifying the email address to send to -->

```

```

<a id="MailTo" xlink:href="mailto:nigel_mck@email.com">

```

```

<rect x="0" y="15" width="80" height="30" style="fill:#FEFEFE; stroke:black; stroke-
width:0.05"/>

```

```

<!-- Animation to lift the lid of the envelope when the mouse moves over -->

```

```

<path id="LiftLid" d="M0,15 L40,0 80,15" style="fill:#FFFFFF;
stroke:black; stroke-width:0.1"
visibility="hidden" pointer-events="none">

```

```

<animate begin="MailTo.mouseover" attributeName="visibility" from="hidden"
to="visible" dur="0.1s" fill="freeze"/>

```

```

<animate begin="MailTo.mouseout" attributeName="visibility" from="visible"
to="hidden" dur="0.1s" fill="freeze"/>
</path>

```

```

<line id="LeftLine" x1="0" y1="15" x2="40" y2="30" style="stroke:black;
stroke-width:0.05" pointer-events="none">

```

```

    <animate begin="MailTo.mouseover" attributeName="y2" from="30" to="0"
dur="0.2s" fill="freeze"/>

```

```

    <animate begin="MailTo.mouseout" attributeName="y2" from="0" to="30" dur="0.1s"
fill="freeze"/>
</line>

```

```

<line id="RightLine" x1="40" y1="30" x2="80" y2="15" style="stroke:black;
stroke-width:0.05" pointer-events="none">

```

```

    <animate begin="MailTo.mouseover" attributeName="y1" from="30" to="0" dur="0.2s"
fill="freeze"/>

```

```

    <animate begin="MailTo.mouseout" attributeName="y1" from="0" to="30" dur="0.1s"
fill="freeze"/>
</line>

```

```

<!-- Animation to increase/decrease the size of the text within the envelope -->

<text x="40" y="40" style="text-anchor:middle; font-size:8; fill:blue; font-family:Arial, sans-serif; pointer-events="none" visibility="hidden">
<animate id="TextUp" begin="MailTo.mouseover" attributeName="y" from="40" to="10"
dur="0.5s" fill="freeze"/>
<animate id="TextDown" begin="MailTo.mouseout" attributeName="y" from="10" to="40"
dur="0.01s" fill="freeze"/>
<animate begin="MailTo.mouseover" attributeName="visibility" from="hidden"
to="visible" dur="0.1s" fill="freeze"/>
<animate begin="MailTo.mouseout" attributeName="visibility" from="visible"
to="hidden" dur="0.01s" fill="freeze"/>
<animate id="TextDownAgain" begin="TextUp.end+0.1s" attributeName="y" from="10"
to="40"
dur="0.5s" fill="freeze"/>
<animate id="TextGrow" begin="TextUp.end+0.1s" attributeName="font-size" from="8"
to="16"
dur="0.5s" fill="freeze"/>
<animate id="TextShrink" begin="MailTo.mouseout" attributeName="font-size" from="16"
to="8"
dur="0.01s" fill="freeze"/>
Email
</text>
</a>
</svg>

</svg>

//*****
/** HTML PAGE WITH A SMIL FILE EMBEDDED WITHIN IT **/
<!-- SMILPresentation.html -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>By Nigel McKelvey 2002 - SMIL Presentation</TITLE>

</HEAD>
<BODY BGCOLOR="beige" text="blue" face="Arial">
<center><h4>SMIL Presentation</h4></center>

<p align="center">
<object id="media"
classid="CLSID:CFCDA03-8BE4-11CF-B84B-0020AFBCCFA" width="550"
height="380">
  <param name="src" value="main.smi">
  <param name="console" value="Clip1">
  <param name="controls" value="ImageWindow">
  <param name="AutoStart" value="TRUE">
  <embed controls="ImageWindow" console="Clip1"
type="audio/x-pn-realaudio-plugin"
src="main.smi"
width="250" height="200" autostart="true">
  </embed>

</object>
<center>

<!-- Embed an SVG link -->

```

```

<EMBED SRC="HomeLink.svg" NAME="SVGEmbed" HEIGHT="33" WIDTH="97"
TYPE="image/svg+xml" PLUGINSPPAGE="http://www.adobe.com/svg/viewer/install/">

<FORM ACTION="bogus_submit.html" METHOD="POST">
  <INPUT NAME="data" TYPE="hidden">
</FORM>
</center>
</BODY>
</HTML>

//*****
/** SMIL FILE - REFERENCES AUDIO, IMAGE AND VIDEO FILES **/
<!-- main.smi -->
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">

  <head>

    <meta name="title" content="SMIL Presentation" />
    <meta name="author" content="CBT Application" />
    <meta name="copyright" content="(c) Nigel McKelvey 2002" />
    <meta name="abstract" content="A SMIL Tutorial on SMIL." />

  <!-- Position all the media in different parts of the screen -->

  <layout>
    <root-layout width="500" height="400" backgroundColor="black" />
    <region id="flash" width="400" height="200" left="50" top="100" z-index="1" />
    <region id="rp" width="196" height="152" left="25" top="50" z-index="4" />
    <region id="vid" width="196" height="152" left="150" top="125" z-index="3" />
    <region id="img" width="196" height="152" left="275" top="200" z-index="2" />
    <region id="logo" width="170" height="81" left="300" top="40" z-index="5" />
    <region id="text" width="285" height="167" left="25" top="210" z-index="1" />
    <regPoint id="centered" regAlign="center" left="50%" top="50%" />
  </layout>

  <!-- 'Sliding' effect of the screens coming in -->

    <transition id="upSlide" type="slideWipe" subtype="fromBottom" />
    <transition id="ellipse" type="ellipseWipe" subtype="horizontal" />
    <transition id="rightSlide" type="slideWipe" subtype="fromRight" />
    <transition id="leftSlide" type="slideWipe" subtype="fromLeft" />
    <transition id="fadeBrown" type="fade" subtype="fadeToColor" fadeColor="#CC9966"/>
    <transition id="fadeOrange" type="fade" subtype="fadeToColor" fadeColor="#FFCC66"/>

  </head>
  <body>

  <!--Assign colours to the screens and display the text from text.rt -->

  <par dur="indefinite">

    <brush id="rp1" color="#FFFFCC" transIn="leftSlide" begin="00:04.5" region="rp"
fill="hold" />
    <brush id="vid1" color="#FFCC66" transIn="leftSlide" begin="00:05.5" region="vid"
fill="hold" />
    <brush id="img1" color="#CC9966" transIn="leftSlide" begin="00:06.5" region="img"
fill="hold" />

```

```

    
    <textstream src="text.rt" begin="00:03.0" region="text" fill="hold" />

```

```
<excl dur="indefinite">
```

```
<!-- Assign images to the screens -->
```

```

<par begin="00:09.0" >
    
        <param name="bitrate" value="1765"/>
    </img>
    
        <param name="bitrate" value="1765"/>
    </img>
    
        <param name="bitrate" value="1765" />
    </img>
</par>

```

```

<par id="yellow" begin="rp1.activateEvent">
    <set targetElement="rp" attributeName="z-index" to="4"/>
    <set targetElement="vid" attributeName="z-index" to="2"/>
    <set targetElement="img" attributeName="z-index" to="3"/>
    

```

```

<!-- Play the audio in conjunction with the -->
<!-- 'slides' of information -->

```

```

<audio src="audio/LearnSMIL.au" />
<seq end="rp1.activateEvent; rp1.activateEvent">
    
        <param name="bitrate" value="1765"/>
    </img>
    
        <param name="bitrate" value="1765"/>
    </img>
    
        <param name="bitrate" value="1765"/>
    </img>
    
        <param name="bitrate" value="1765"/>
    </img>
    
        <param name="bitrate" value="1765"/>
    </img>
    
        <param name="bitrate" value="1765"/>
    </img>

```



```

        <brush color="#CC9966" region="rp" fill="hold"/>
        
            <param name="bitrate" value="1765" />
        </img>
    </seq>
</par>

<par id="orange" begin="vid1.activateEvent">
    <set targetElement="vid" attributeName="z-index" to="4"/>
    <set targetElement="rp" attributeName="z-index" to="2"/>
    <set targetElement="img" attributeName="z-index" to="3"/>

    <!-- Play the video -->
    <!-- considering the bitrate -->

    <video src="video/nigel.avi" region="vid" transIn="ellipse" transOut="fadeOrange"
    regPoint="centered" fill="remove"/>
        <brush color="#CC9966" region="img" fill="hold"/>
        
            <param name="bitrate" value="1765" />
        </img>

        <brush color="#FFFFCC" region="rp" fill="hold"/>
        
            <param name="bitrate" value="1765" />
        </img>
    </par>

<par id="brown" begin="img1.activateEvent">
    <set targetElement="rp" attributeName="z-index" to="2"/>
    <set targetElement="vid" attributeName="z-index" to="3"/>
    <set targetElement="img" attributeName="z-index" to="4"/>

    <!-- Play the audio in conjunction with the -->
    <!-- 'slides' of information -->

    <audio src="audio/LearnSMIL.au" />
    <seq end="vid1.activateEvent; rp1.activateEvent">
        
            <param name="bitrate" value="1765"/>
        </img>
        
            <param name="bitrate" value="1765"/>
        </img>

        
            <param name="bitrate" value="1765"/>
        </img>
        
            <param name="bitrate" value="1765"/>
        </img>
    </seq>
</par>

```



```

        
        <param name="bitrate" value="1765"/>
    </img>
        
        <param name="bitrate" value="1765"/>
    </img>
        <brush color="#CC9966" region="img" fill="hold"/>
        
        <param name="bitrate" value="1765" />
    </img>
    <brush color="#FFCC66" region="vid" fill="hold"/>
        
        <param name="bitrate" value="1765" />
    </img>
    <brush color="#FFFFCC" region="rp" fill="hold"/>
        
        <param name="bitrate" value="1765" />
    </img>
</seq>
</par>
</excl>
</par>
</body>
</smil>

```

```

//*****
/** HTML PAGE WITH SVG EMBEDDED WITHIN IT **/
<!-- coll.html -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>By Nigel McKelvey 2002</TITLE>
</HEAD>
<BODY BGCOLOR="beige">
<EMBED SRC="cool2_2.svg" NAME="SVGEmbed" HEIGHT="440" WIDTH="780"
TYPE="image/svg+xml" PLUGINSPPAGE="http://www.adobe.com/svg/viewer/install/">

<FORM ACTION="bogus_submit.html" METHOD="POST">
    <INPUT NAME="data" TYPE="hidden">
</FORM>

</BODY>
</HTML>

```

```

//*****
/** SVG PAGE - ALL TEXT, LINKS, GRAPHICS, ETC. ARE **/
/** CREATED USING SVG **/
<!-- cool2_2.svg -->
<?xml version='1.0'?>
<!-- Created in November 2002 by Nigel McKelvey at LYIT-->
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20010904//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<!-- For the moment mimic an 800 x 600 display -->
<svg width="100%" height="100%"
    xmlns:xlink="http://www.w3.org/1999/xlink"

```

```

xmlns = 'http://www.w3.org/2000/svg'>

<title>Nigel McKelvey (c) 2002</title>

<!-- TOP PART OF THE SCREEN -->
<rect x="0" y="0" width="1200" height="100" style="fill:#999999;"/>
<svg x="166" y="20" width="468" height="60">
<text x="700" y="40" style="stroke:black; fill:blue; font-family:'Times New
Roman', serif; font-size:24; font-weight:normal;">
Tutorial on XML Applications written entirely in SVG !

<animate attributeName="x" from="650" to="-900" begin="0s" dur="20s"
repeatCount="indefinite"/>
</text>
<rect x="0" y="0" width="468" height="60" style="stroke:beige;
stroke-width:2; fill:none;"/>
</svg>

<!-- Left "frame" -->
<rect x="0" y="100" width="100" height="700" style="fill:beige;"/>
<a id="SVG">
<text style="fill:black; stroke:black; font-size:12;" x="25" y="130" >
SVG
</text>
</a>
<a id="SMIL">
<text style="fill:black; stroke:black; font-size:12;" x="25" y="160" >
SMIL
</text>
</a>
<a id="MathML">
<text style="fill:black; stroke:black; font-size:12;" x="25" y="190" >
MathML
</text>
</a>
<a id="VML">
<text style="fill:black; stroke:black; font-size:12;" x="25" y="220" >
VML
</text>
</a>
<a id="XML">
<text style="fill:black; stroke:black; font-size:12;" x="25" y="250" >
XML
</text>
</a>
<a id="Support">
<text style="fill:black; stroke:black; font-size:12;" x="25" y="280" >
Support
</text>
</a>
<a id="Conclusions">
<text style="fill:black; stroke:black; font-size:12;" x="25" y="310" >
Conclusions
</text>
</a>

<!-- Main Frame -->
<rect style="fill:#CCCCCC;" x="100" y="100" width="1100" height="700"/>

```

```

<rect style="fill:black;" x="100" y="100" width="2" height="700"/>
<text x="130" y="130" style="font-family:Arial, sans-serif; font-size:14;
font-weight:normal;">
This site was created entirely using SVG code. Creating graphics is a lot
easier than making text!</text>
<text>
<tspan x="130" y="160" style="font-family:Arial, sans-serif; font-size:14;
font-weight:normal;">
As you can see on the left hand side, a menu is placed for your convenience
that will allow you
</tspan>
<tspan x="130" dy="1em" style="font-family:Arial,sans-serif; font-size:14;
font-weight:normal;">
too see a brief summary of what is stored on every page. Each page itself is
made using SVG code.
</tspan>
<tspan x="130" dy="2em" style="font-family:Arial,sans-serif; font-size:14;
font-weight:normal;">
SVG is a powerful language in that it can dramatically reduce file
sizes for use on the Net.
</tspan>
<tspan x="130" dy="2em" style="font-family:Arial,sans-serif; font-size:14;
font-weight:normal;">
The tutorials available here will give the user a better understanding of what
is involved in creating
</tspan>
<tspan x="130" dy="1em" style="font-family:Arial,sans-serif; font-size:14;
font-weight:normal;">
applications that are derived from XML. The 'X' in XML is representative
of a sense of
</tspan>
<tspan x="130" dy="1em" style="font-family:Arial,sans-serif; font-size:14;
font-weight:normal;">
eXtensibility, which is required for modern applications. Sematic information
for example.
</tspan>
<tspan x="130" dy="2em" style="font-family:Arial,sans-serif; font-size:14;
font-weight:normal;">
XML is a relatively new language which adds added functionality to many
aspects of the Internet.
</tspan>
<tspan x="130" dy="1em" style="font-family:Arial,sans-serif; font-size:14;
font-weight:normal;">
Some of these languages are discussed in this tutorial such as SVG,
SMIL, MathML, SVG,
</tspan>
<tspan x="130" dy="1em" style="font-family:Arial,sans-serif; font-size:14;
font-weight:normal;"> .
VML etc.
</tspan>

<tspan x="130" dy="2em" style="font-family:Arial,sans-serif; font-size:14;
font-weight:normal;">
Enjoy the tutorials!
</tspan>

<tspan x="130" dy="2em" style="stroke:blue; fill:yellow; font-family:Arial,
sans-serif; font-size:14; font-weight:normal;">

```

```
Email Nigel Mc Kelvey at: <a
xlink:href="mailto:nigelmckelvey@lyit.ie"><tspan style="stroke:black; fill:blue">
Nigel McKelvey (LYIT)</tspan></a>
</tspan>
```

```
<tspan x="130" dy="2em" style="font-family:Arial,sans-serif; font-size:14;
font-weight:normal; stroke:yellow; fill:blue;">
<tspan style="stroke:black; fill:blue;">
<a xlink:href="X:\Menu\openingScreen3_2.html" id="HomePage">Home Page</a></tspan>
</tspan>
```

```
</text>
```

```
<!-- The following are the labels to be displayed when a link is moused. -->
```

```
<svg x="80" y="140" id="LinkInfo" width="200" height="150" visibility="hidden">
<animate begin="HomePage.mouseover" dur="0.1s" attributeName="visibility"
from="hidden" to="visible" fill="freeze"/>
<animate begin="HomePage.mouseout" dur="0.1s" attributeName="visibility"
from="visible" to="hidden" fill="freeze"/>
<rect x="0" y="0" width="100%" height="25" style="fill:#EEEEEE; stroke:blue;
stroke-width:2"/>
<rect x="0" y="25" width="100%" height="125" style="fill:#CCCCCC; stroke:blue;
stroke-width:2,"/>
<text>
<tspan x="10" y="20" style="fill:blue; font-size:16; font-family: Arial,
sans-serif" >
Home Page
</tspan>
<tspan x="10" dy="2em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
The Home Page shows a clock
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
that is created using both
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
SVG code and JavaScript!
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
Links are also provided here
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
to other sections of the
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
application.
</tspan>
</text>
</svg> <!-- Ends the information on the Home Page. -->
```

```
<svg x="80" y="140" width="200" height="150" visibility="hidden">
<animate begin="SVG.mouseover" dur="0.1s" attributeName="visibility"
```

```

from="hidden" to="visible" fill="freeze"/>
<animate begin="SVG.mouseout" dur="0.1s" attributeName="visibility"
from="visible" to="hidden" fill="freeze"/>
<rect x="0" y="0" width="100%" height="25" style="fill:#EEEEEE;
stroke:blue;stroke-width:2"/>
<rect x="0" y="25" width="100%" height="125" style="fill:#CCCCCC; stroke:blue;
stroke-width:2"/>
<text>
<tspan x="10" y="20" style="fill:blue; font-size:16; font-family: Arial,
sans-serif" >
SVG
</tspan>
<tspan x="10" dy="2em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
The SVG page demonstrates
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
some of the applications
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
available, giving a
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
tutorial and code samples.
</tspan>

</text>
</svg> <!-- Ends the information about SVG. -->

<svg x="80" y="170" id="LinkInfo" width="200" height="150" visibility="hidden">
<animate begin="SMIL.mouseover" dur="0.1s" attributeName="visibility"
from="hidden" to="visible" fill="freeze"/>
<animate begin="SMIL.mouseout" dur="0.1s" attributeName="visibility"
from="visible" to="hidden" fill="freeze"/>
<rect x="0" y="0" width="100%" height="25" style="fill:#EEEEEE; stroke:blue;
stroke-width:2"/>
<rect x="0" y="25" width="100%" height="125" style="fill:#CCCCCC; stroke:blue;
stroke-width:2;"/>
<text>
<tspan x="10" y="20" style="fill:blue; font-size:16; font-family: Arial,
sans-serif" >
SMIL
</tspan>
<tspan x="10" dy="2em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
Synchronised Multimedia
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
Integration Language is
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
described on this page
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">

```

```

giving background theory
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
and sample code.
</tspan>

</text>
</svg> <!-- Ends the information on the SMIL. -->

<svg x="80" y="200" id="LinkInfo" width="200" height="150" visibility="hidden">
<animate begin="MathML.mouseover" dur="0.1s" attributeName="visibility"
from="hidden" to="visible" fill="freeze"/>
<animate begin="MathML.mouseout" dur="0.1s" attributeName="visibility"
from="visible" to="hidden" fill="freeze"/>
<rect x="0" y="0" width="100%" height="25" style="fill:#EEEEEE; stroke:blue;
stroke-width:2"/>
<rect x="0" y="25" width="100%" height="125" style="fill:#CCCCCC; stroke:blue;
stroke-width:2;"/>
<text>
<tspan x="10" y="20" style="fill:blue; font-size:16; font-family: Arial,
sans-serif" >
MathML
</tspan>
<tspan x="10" dy="2em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
On this page MathML is
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
discussed with particular
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
reference to the reasons
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
for its creation including
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
code samples.
</tspan>

</text>
</svg> <!-- Ends the information on the MathML. -->

<svg x="80" y="230" id="LinkInfo" width="200" height="150" visibility="hidden">
<animate begin="VML.mouseover" dur="0.1s" attributeName="visibility"
from="hidden" to="visible" fill="freeze"/>
<animate begin="VML.mouseout" dur="0.1s" attributeName="visibility"
from="visible" to="hidden" fill="freeze"/>
<rect x="0" y="0" width="100%" height="25" style="fill:#EEEEEE; stroke:blue;
stroke-width:2"/>
<rect x="0" y="25" width="100%" height="125" style="fill:#CCCCCC; stroke:blue;
stroke-width:2;"/>
<text>
<tspan x="10" y="20" style="fill:blue; font-size:16; font-family: Arial,
sans-serif" >

```



```

VML
</tspan>
<tspan x="10" dy="2em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
Here a discussion is given
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
on the differences and
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
similarities between SVG and
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
VML. Research carried out
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
will also be backed up
</tspan>

</text>
</svg> <!-- Ends the information on the VML. -->

<svg x="80" y="260" id="LinkInfo" width="200" height="150" visibility="hidden">
<animate begin="XML.mouseover" dur="0.1s" attributeName="visibility"
from="hidden" to="visible" fill="freeze"/>
<animate begin="XML.mouseout" dur="0.1s" attributeName="visibility"
from="visible" to="hidden" fill="freeze"/>
<rect x="0" y="0" width="100%" height="25" style="fill:#EEEEEE; stroke:blue;
stroke-width:2"/>
<rect x="0" y="25" width="100%" height="125" style="fill:#CCCCCC; stroke:blue;
stroke-width:2"/>
<text>
<tspan x="10" y="20" style="fill:blue; font-size:16; font-family: Arial,
sans-serif" >
XML
</tspan>
<tspan x="10" dy="2em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
This page will demonstrate
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
the capabilities of XML,
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
its applications and future.
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
Code samples will also be given
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
showing working examples.
</tspan>

```



```

</text>
</svg> <!-- Ends the information on the XML. -->

<svg x="80" y="290" id="LinkInfo" width="200" height="150" visibility="hidden">
<animate begin="Support.mouseover" dur="0.1s" attributeName="visibility"
from="hidden" to="visible" fill="freeze"/>
<animate begin="Support.mouseout" dur="0.1s" attributeName="visibility"
from="visible" to="hidden" fill="freeze"/>
<rect x="0" y="0" width="100%" height="25" style="fill:#EEEEEE; stroke:blue;
stroke-width:2"/>
<rect x="0" y="25" width="100%" height="125" style="fill:#CCCCCC; stroke:blue;
stroke-width:2;"/>
<text>
<tspan x="10" y="20" style="fill:blue; font-size:16; font-family: Arial,
sans-serif" >
Support
</tspan>
<tspan x="10" dy="2em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
This is a very important
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
page as it lists the support
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
that is currently available
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
for XML applications (ie:
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
OS, Browsers etc)
</tspan>

</text>
</svg> <!-- Ends the information on the Support. -->

<svg x="80" y="320" id="LinkInfo" width="200" height="150" visibility="hidden">
<animate begin="Conclusions.mouseover" dur="0.1s" attributeName="visibility"
from="hidden" to="visible" fill="freeze"/>
<animate begin="Conclusions.mouseout" dur="0.1s" attributeName="visibility"
from="visible" to="hidden" fill="freeze"/>
<rect x="0" y="0" width="100%" height="25" style="fill:#EEEEEE; stroke:blue;
stroke-width:2"/>
<rect x="0" y="25" width="100%" height="125" style="fill:#CCCCCC; stroke:blue;
stroke-width:2;"/>
<text>
<tspan x="10" y="20" style="fill:blue; font-size:16; font-family: Arial,
sans-serif" >
Conclusions
</tspan>
<tspan x="10" dy="2em" style="fill:black; font-size:12; font-family: Arial,
sans-serif">
As a conclusion to the tutorial
</tspan>
<tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,

```

```

    sans-serif">
    a brief synopsis of the various
  </tspan>
  <tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
  sans-serif">
  applications will be given with
  </tspan>
  <tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
  sans-serif">
  a line or two on how I see the
  </tspan>
  <tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
  sans-serif">
  applications progressing in the
  </tspan>
  <tspan x="10" dy="1.5em" style="fill:black; font-size:12; font-family: Arial,
  sans-serif">
  future. FREE!
  </tspan>
</text>
</svg> <!-- Ends the information on the Conclusions. -->
</svg>

/*****
/** HTML PAGE THAT EMBEDS AN XML FILE AS AN      **/
/** ISLAND OF DATA                               **/
<!-- index.htm -->
<html>

<!-- ***** -->
<!-- Nigel McKelvey                               -->
<!-- 2003                                         -->
<!-- This system demonstrates the uses of Data Islands. -->
<!-- An XML document is embedded in the code of this -->
<!-- HTML page and its contents is used for display -->
<!-- and manipulation purposes. Thus demonstrating -->
<!-- that XML can be regarded as a database in itself -->
<!-- ***** -->

<body onload="loadCombo();" bgcolor="beige">

  <!-- Embed the XML document -->

  <xml id="sourceXML" src="phonebook.xml"></xml>
  <xml id="resultXML"></xml>
  <center>
  <table>
  <tr><td><span id="cboNames"></span></td>
  <td>
    <span class="clsData">| FirstName</span>
    <input class="clsData" id="srchfname" size="10">
    <span class="clsData">Last-Name</span>
    <input class="clsData" id="srchlname" size="10" >

    <!-- Call on the function to search for the name -->

    <input type="button" value="Search" onClick="srchOnName()" class="clsData" >

  </td>
  </tr>

```

```

</table>

<div id="divResult" style="BACKGROUND-COLOR: lemonchiffon; BORDER-BOTTOM-STYLE:
double; BORDER-LEFT-STYLE: double; BORDER-RIGHT-STYLE: double; BORDER-TOP-
STYLE: double">
<table datasrc="#resultXML">

<thead bgcolor="lightsteelblue">
  <TR>
    <th>Employee ID#</th>
    <th>Employee Type</th>
    <th>First Name</th>
    <th>Middle Name</th>
    <th>Last Name</th>
    <th>Nickname</th>
    <th>Contact Info:</th>
  </TR>
</thead>

<tr>

<!-- tag names of the information in the XML document -->

<td><span datafld="empid"></span></td>
<td><span datafld="emptype"></span></td>
<td><span datafld="fname"></span></td>
<td><span datafld="mname"></span></td>
<td><span datafld="lname"></span></td>
<td><span datafld="nickname"></span></td>
<td>
  <table border="1" datasrc="#resultXML" datafld="contactentry">
    <tr><td><span datafld="type"></span></td><td><span datafld="data"></span></td></tr>
  </table>
</td>
</tr>

</table>

</div>

<!-- Embed an SVG link -->

<EMBED SRC="x:\menu\HomeLink.svg" NAME="SVGEmbed" HEIGHT="33" WIDTH="97"
TYPE="image/svg+xml" PLUGINSPPAGE="http://www.adobe.com/svg/viewer/install/">

<FORM ACTION="bogus_submit.html" METHOD="POST">
  <INPUT NAME="data" TYPE="hidden">
</FORM>
</center>

<!-- Java Script required in order to 'search' the 'DB' -->
<!-- and to load the combo. -->

<script language="JavaScript">
function loadCombo()
{
  var nl = sourceXML.XMLDocument.documentElement.selectNodes("person")
  var vhtml ="";
  var n =""
  var ln =""

```

```

var fn = ""

/* creates a combo box puts in the first and last names as values */
for (var i=0;i<nl.length;i++)
{
    n = nl(i)
    ln = n.attributes.getNamedItem("lname").value
    fn = n.attributes.getNamedItem("fname").value
    vhtml+= '<option value="' + ln + '"' + fn + '">' + ln + ", " + fn + '</option>\r'
}

cboNames.outerHTML = '<select id="cboNames" onchange="cboChange();">' + vhtml +
'</select>';
cboNames.selectedIndex = -1;
}

function cboChange()
{
    /*retrieves the first and last name from the combo into an array */
    var nm = cboNames.options(cboNames.selectedIndex).value.split("|")
    srchfname.value = nm[1];
    srchlname.value = nm[0];
    nm="";
}

function srchOnName()
{
    /*make sure at least one of the input boxes are filled in*/
    if(srchfname.value.length !=0 || srchlname.value.length !=0)
    {

        /*create a local DOM*/
        var x = new ActiveXObject("MSXML.DOMDocument");
        x.loadXML("<PHONEBOOK/>")

        /*check to see if both inputs are filled*/
        if(srchfname.value.length !=0 && srchlname.value.length !=0)
        {
            var
            nl=sourceXML.XMLDocument.documentElement.selectNodes("person[@lname=" +
            srchlname.value + " and @fname=" + srchfname.value + " ]");
        }
        else {
            /*check to see which one of the inputs are blank*/
            if(srchfname.value.length ==0)
            {
                /*search by last name*/
                var
                nl=sourceXML.XMLDocument.documentElement.selectNodes("pe
                rson[@lname=" + srchlname.value + " ]")
            }
            else
            {
                /*search by first name*/
                var
                nl=sourceXML.XMLDocument.documentElement.selectNodes("pe
                rson[@fname=" + srchfname.value + " ]")
            }
        }
    }
}

```

```

    }

    if(nl)
    {
        for(var i =0;i < nl.length;i++)
        {
            /*append a copy of each node found*/
            x.documentElement.appendChild(nl(i).cloneNode(true))
        }
        /*load results*/
        resultXML.XMLDocument.loadXML(x.xml);
    }

    x="";
    nl="";
}
}
</script>
</body>
</html>

```

```

/*****
** THE XML FILE THAT ACTS AS THE 'ISLAND OF DATA' **
<!-- phonebook.xml -->
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="search.xsl"?>

<PHONEBOOK>
  <person
    empid="6155" emptype="E"
    fname="Eimear" mname="" lname="Alexander"
    nickname="" >
    <contactentry type="Office" data="074-49654" />
    <contactentry type="Pager" data="888-948-6359" />
  </person>
  <person
    empid="6368" emptype="E"
    fname="Michelle" mname="E" lname="Bonner"
    nickname="" >
    <contactentry type="Office" data="074-9362222" />
    <contactentry type="Mobile" data="086-3426343" />
  </person>
  <person
    empid="6083" emptype="E"
    fname="Richard" mname="E" lname="Conaghan"
    nickname="" >
    <contactentry type="Office" data="513-573-4704" />
  </person>
  <person
    empid="6035" emptype="E"
    fname="Angela" mname="M" lname="Doherty"
    nickname="" >
    <contactentry type="Office" data="074-49104" />
    <contactentry type="Mobile" data="086-8877654" />
    <contactentry type="Pager" data="888-948-6344" />
  </person>
  <person
    empid="5818" emptype="E"
    fname="Ed" mname="J" lname="Egan"

```

```

    nickname="" >
      <contactentry type="Office" data="513-573-7429" />
    </person>
  <person
    empid="6060" emptytype="E"
    fname="F" mname="Scott" lname="Fitzgerald"
    nickname="" >
      <contactentry type="Office" data="513-573-4484" />
      <contactentry type="Mobile" data="206-947-3720" />
    </person>
  <person
    empid="6042" emptytype="E"
    fname="Michael" mname="J" lname="Gallagher"
    nickname="" >
      <contactentry type="Office" data="1800-7865433" />
    </person>
  <person
    empid="5626" emptytype="E"
    fname="Damien" mname="C" lname="Harvey"
    nickname="sleepy" >
      <contactentry type="Office" data="074-49876" />
      <contactentry type="Mobile" data="085-5645342" />
      <contactentry type="Pager" data="888-948-6346" />
    </person>
  <person
    empid="5905" emptytype="Dr"
    fname="John" mname="O" lname="Inderhaug"
    nickname="Weigen" >
      <contactentry type="Pager" data="678-987-5" />
    </person>
  <person
    empid="6119" emptytype="E"
    fname="John" mname="P" lname="Jones"
    nickname="" >
      <contactentry type="Office" data="074-31321" />
      <contactentry type="Pager" data="888-948-9251" />
    </person>
  <person
    empid="5965" emptytype="E"
    fname="David" mname="E" lname="Kelly"
    nickname="" >
      <contactentry type="Office" data="073-67453" />
      <contactentry type="Mobile" data="086-7856454" />
      <contactentry type="Pager" data="888-876-7220" />
    </person>
  <person
    empid="5512" emptytype="E"
    fname="Molly" mname="R" lname="Lavelle"
    nickname="" >
      <contactentry type="Home" data="074-9363402" />
      <contactentry type="Mobile" data="087-665453" />
    </person>
  <person
    empid="5890" emptytype="E"
    fname="Ruth" mname="G" lname="Lennon"
    nickname="" >
      <contactentry type="Office" data="074-64313" />
      <contactentry type="Pager" data="999" />
    </person>
  <person

```

```

empid="4696" emptytype="E"
fname="Nigel" mname="A" lname="McKelvey"
nickname="" >
  <contactentry type="Office" data="074-64545" />
  <contactentry type="Mobile" data="087-6636014" />
</person>
<person
empid="4088" emptytype="E"
fname="William" mname="A" lname="Nellis"
nickname="" >
  <contactentry type="Office" data="513-573-7409" />
  <contactentry type="Mobile" data="513-310-6881" />
</person>
<person
empid="1742" emptytype="E"
fname="Steve" mname="G" lname="OBoyle"
nickname="" >
  <contactentry type="Office" data="513-573-4496" />
</person>
<person
empid="10140" emptytype="C"
fname="Jamie" mname="E" lname="Patton"
nickname="" >
  <contactentry type="Office" data="513-573-4511" />
</person>
<person
empid="5720" emptytype="E"
fname="Phyllis" mname="A" lname="Quinn"
nickname="" >
  <contactentry type="Office" data="513-573-7273" />
</person>
<person
empid="10038" emptytype="C"
fname="Jim" mname="R" lname="Rushe"
nickname="" >
  <contactentry type="Office" data="513-573-4463" />
</person>
<person
empid="10097" emptytype="C"
fname="John" mname="R" lname="Scott"
nickname="" >
  <contactentry type="Office" data="513-573-7339" />
</person>
<person
empid="5681" emptytype="E"
fname="Ed" mname="A" lname="Tinney"
nickname="" >
  <contactentry type="Office" data="513-573-7432" />
  <contactentry type="Pager" data="888-876-7317" />
</person>
<person
empid="10100" emptytype="C"
fname="Holly" mname="C" lname="Verry"
nickname="" >
  <contactentry type="Office" data="513-573-5475" />
  <contactentry type="Pager" data="513-650-3390" />
  <contactentry type="Mobile" data="513-617-3149" />
</person>
<person
empid="5904" emptytype="E"

```



```

    fname="Lara" mname="M" lname="Wallace"
    nickname="" >
      <contactentry type="Office" data="513-573-7459" />
      <contactentry type="Pager" data="888-876-7331" />
    </person>
  <person
    empid="5862" emptype="E"
    fname="Alex" mname="M" lname="Wilton"
    nickname="" >
      <contactentry type="Office" data="513-573-7378" />
      <contactentry type="Pager" data="888-876-7239" />
    </person>
  <person
    empid="5719" emptype="E"
    fname="Tom" mname="A" lname="Williams"
    nickname="" >
      <contactentry type="Office" data="513-573-7271" />
    </person>
  <person
    empid="5986" emptype="E"
    fname="Tom" mname="F" lname="Williamson"
    nickname="" >
      <contactentry type="Office" data="513-573-4504" />
      <contactentry type="Pager" data="888-356-8662" />
    </person>
</PHONEBOOK>

/*****
** HTML FILE THAT IMPLEMENTS XML, CSS, ETC. IN      **/
** ORDER TO DISPLAY INFORMATION IN TABLE FORMAT **/
<!-- Default.htm -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html><HEAD>
  <title>Search through the Personal Database - Implementing XSL/CSS</title>

<!-- Embed an SVG Link -->

<center>
  <EMBED SRC="x:\menu\HomeLink.svg" NAME="SVGEmbed" HEIGHT="33" WIDTH="97"
  TYPE="image/svg+xml" PLUGINSPPAGE="http://www.adobe.com/svg/viewer/install/">

  <FORM ACTION="bogus_submit.html" METHOD="POST">
    <INPUT NAME="data" TYPE="hidden">
  </FORM>
</center>

<!-- Reference a CSS file to format the table etc. -->

  <link href="Database.css" rel="stylesheet">

  <body bgcolor="beige"> <center>
    <form method="post" action="" name="frmData">
      <div id="divHeader" style="Z-INDEX: 34; LEFT: 33px; WIDTH: 703px;
      POSITION: absolute; TOP: 324px; HEIGHT: 150px">

        <div name="PeopleList" style="LEFT: 1px; OVERFLOW: scroll; WIDTH:
        100%; TOP: 14px; HEIGHT: 125px">

```

```

<table id="tblHeader" Datasrc="#xmlDSO" Class="DataTable"
width="100%" cellpadding="0" cellspacing="0" border="1" style="LEFT: -
1px; TOP: 0px; ">
  <thead>
    <tr>
      <th id="thdr" width="10" SortField="ID" class =
"SortingGridUp">#</th>
      <th id="thdr" width="80" SortField="FullName"
class = "SortingGridUp">Name</th>
      <th id="thdr" width="250"
SortField="Description" class =
"SortingGridUp">Description</th>
      <th id="thdr" width="49" SortField="Age" class
= "SortingGridUp">Age</th>
      <th id="thdr" width="67" SortField="Location"
class = "SortingGridUp">Location</th>
      <th id="thdr" width="90"
SortField="Occupation" class =
"SortingGridUp">Occupation</th>
      <th id="thdr" width="139" SortField="Hobbies"
class = "SortingGridUp" >Hobbies</th>
    </tr>
  </thead>
  <tbody>
    <tr onclick="CurrentRecord()">
      <td width="10" align="left" height="15"><span
datafld="ID" class="xDataCol"></span></td>
      <td width="80" align="left" height="15"><span
datafld="FullName"
class="xDataCol"></span></td>
      <td width="250" align="left" height="15"><span
datafld="Description"
class="xDataCol"></span></td>
      <td width="49" align="left" height="15"><span
datafld="Age" class="xDataCol"></span></td>
      <td width="67" align="left" height="15"><span
datafld="Location"
class="xDataCol"></span></td>
      <td width="90" align="left" height="15"><span
datafld="Occupation"
class="xDataCol"></span></td>
      <td width="139" align="left" height="15"><span
datafld="Hobbies"
class="xDataCol"></span></td>
    </tr>
  </tbody>
</table>
</div>

<div id="main" style="LEFT: 0px; WIDTH: 759px; POSITION: absolute; TOP: 410px; HEIGHT:
29px" tabIndex="11">
  <div id="divSearch" style="LEFT: 32px; WIDTH: 710px; POSITION: absolute; TOP: -
255px; HEIGHT: 27px">
    <table ID="tblSearchFields" border=1 bgcolor="menu" width="100%">

    </table>

  </div>

```



```
<script language="JavaScript">

function doShowAll()
{
    try
    {
        xmlDSO.loadXML(xmlDSOorig.xml);
    }
    catch (error)
    {

        alert("doShowAll() in Default.htm caused the following error: " + error.description)
    }
}

function doFilter()
{
    try
    {
        var strOccupation = document.all.selOccupation.value;
        var strType = document.all.radType.value;
        var strLocation = document.all.selLocation.value;
        if (document.all.radType(1).checked == true)
        {
            strType = document.all.radType(1).value;
        }
        else
        {
            strType = document.all.radType(0).value;
        }
        vbFilterTable(strLocation, strOccupation, strType);
    }
    catch (error)
    {
        alert("doFilter() in Database.htm caused the following error: " + error.description)
    }
}

function CurrentRecord()
{
    try
    {
        var iNum = window.event.srcElement.recordNumber;
        xmlDSO.recordset.AbsolutePosition = iNum;
    }
    catch (error)
    {
        alert("CurrentRecord() in Database.htm caused the following error: " +
            error.description)
    }
}

</script>
</body>
</html>
```

```

//*****
/** THE XML FILE THAT WILL POPULATE THE TABLE ABOVE **/
<!-- Database.xml -->
<root>
  <data ID="01" FullName="M. Rodgers"
    Description="Never attends class!" Age="19"
    Location="Co. Sligo" Occupation="Student"
    Income="&lt;6,000 p/a" Hobbies="Soccer"
    Type="Male" EmpPhoto="images/Access1.gif"/>
  <data ID="02" FullName="L. Kelly"
    Description="Good Student, works well" Age="24"
    Location="Co. Donegal" Occupation="Student"
    Income="10,000+ p/a" Hobbies="Swimming"
    Type="Female" EmpPhoto="images/Access2.gif"/>
  <data ID="03" FullName="E. Cooke"
    Description="Lecturer in Computing" Age="38"
    Location="Co. Limerick" Occupation="AL"
    Income="30,000 p/a" Hobbies="Failing People"
    Type="Female" EmpPhoto="images/Access3.gif"/>
  <data ID="04" FullName="E. Scott"
    Description="Has progressed well" Age="23"
    Location="Co. Donegal" Occupation="M.Sc. Student"
    Income="7,000 p/a" Hobbies="Socialising"
    Type="Female" EmpPhoto="images/Access4.gif"/>
  <data ID="05" FullName="E. Rowan"
    Description="Very Determined" Age="26"
    Location="Co. Mayo" Occupation="IT Student"
    Income="8,000 p/a" Hobbies="Painting"
    Type="Male" EmpPhoto="images/Access5.gif"/>
  <data ID="06" FullName="M. Bonner"
    Description="Hard worker" Age="23"
    Location="Co. Donegal" Occupation="Lab Technician"
    Income="too much!" Hobbies="Socialising"
    Type="Female" EmpPhoto="images/Access6.gif"/>
  <data ID="07" FullName="D. McKelvey"
    Description="Married with kids" Age="36"
    Location="Co. Donegal" Occupation="Self Employed"
    Income="40,000+ p/a" Hobbies="Drinking!"
    Type="Male" EmpPhoto="images/SMIL1.gif"/>
  <data ID="08" FullName="L. McGinley"
    Description="Needs to improve" Age="22"
    Location="Co. Armagh" Occupation="Computing Student"
    Income="&lt; 5,000 p/a" Hobbies="Music"
    Type="Male" EmpPhoto="images/SMIL2.gif"/>

</root>

//*****
/** THE CSS FILE THAT WILL FORMAT THE DATA STORED IN THE      **/
/** XML FILE ABOVE AND ALSO FORMAT THE TABLE IN WHICH THE    **/
/** DATA WILL BE DISPLAYED                                    **/

<!-- Database.css -->
INPUT
{
  FONT-SIZE: 9pt;
  HEIGHT:20px;
  COLOR: #330066;
  FONT-FAMILY: Arial, Helvetica, sans-serif;
  BACKGROUND-COLOR: #F0F0F0;
}

```

```

        border : thin outset #F5F5DC;
        border-bottom-color : #F5FFFA;
    }

FORM
{
    FONT-SIZE: 10pt;
    COLOR: #330066;
    FONT-FAMILY: Arial, Helvetica, sans-serif
}

TD
{
    FONT-SIZE: 9pt;
    FONT-FAMILY: Arial, Helvetica, sans-serif;
}

BODY
{
    SCROLLBAR-FACE-COLOR: #F0F0F0;
    SCROLLBAR-HIGHLIGHT-COLOR: #CCCCCC;
    SCROLLBAR-SHADOW-COLOR: #CCCCCC;
    SCROLLBAR-3DLIGHT-COLOR: #F0F0F0;
    SCROLLBAR-ARROW-COLOR: #330066;
    SCROLLBAR-DARKSHADOW-COLOR: #CCCCCC;
    SCROLLBAR-BASE-COLOR: #CCCCCC;
}

.SortingGridUp
{
    BORDER-RIGHT: #999999 2px outset;
    BORDER-TOP: #e4ecf5 1px outset;
    FONT-WEIGHT: normal;
    FONT-SIZE: 8pt;
    BORDER-LEFT: #e4ecf5 1px outset;
    CURSOR: hand;
    COLOR: #2d1663;
    BORDER-BOTTOM: #adc3e4 1px outset;
    FONT-FAMILY: Arial, Helvetica, sans-serif;
    HEIGHT: 1px
}

.SortingAscending
{
    BORDER-RIGHT: #e4ecf5 2px inset;
    BACKGROUND-POSITION: right center;
    BORDER-TOP: #adc3e4 1px inset;
    FONT-WEIGHT: normal;
    FONT-SIZE: 8pt;
    BACKGROUND-IMAGE: url(images/UpArrow.gif);
    BORDER-LEFT: #adc3e4 1px inset;
    CURSOR: hand;
    COLOR: #8B0000;
    BORDER-BOTTOM: #e4ecf5 1px inset;
    BACKGROUND-REPEAT: no-repeat;
    FONT-FAMILY: Arial, Helvetica, sans-serif;
    HEIGHT: 1px
}

```

```
.SortingDescending
{
  BORDER-RIGHT: #e4ecf5 2px inset;
  BACKGROUND-POSITION: right center;
  BORDER-TOP: #adc3e4 1px inset;
  FONT-WEIGHT: normal;
  FONT-SIZE: 8pt;
  BACKGROUND-IMAGE: url(images/DownArrow.gif);
  BORDER-LEFT: #adc3e4 1px inset;
  CURSOR: hand;
  COLOR: #8B0000;
  BORDER-BOTTOM: #e4ecf5 1px inset;
  BACKGROUND-REPEAT: no-repeat;
  FONT-FAMILY: Arial, Helvetica, sans-serif;
  HEIGHT: 1px
}

```

```
TR
{
  CURSOR: hand;
  TEXT-ALIGN: left;
  PADDING-LEFT : 4px;
}

```

```
.DataTable
{
  BORDER-RIGHT: 0px solid;
  BORDER-TOP: thin solid;
  FONT-WEIGHT: normal;
  FONT-SIZE: 8pt;
  BORDER-LEFT: 0px solid;
  BORDER-BOTTOM: 1px solid;
  FONT-FAMILY: Arial, Helvetica, sans-serif;
  HEIGHT: 1px
}

```

```
.SortingGridUp
{
  BORDER-RIGHT: highlight inset 1px;
  FONT-WEIGHT: normal;
  FONT-SIZE: 8pt;
  CURSOR: hand;
  COLOR: highlight;
  BACKGROUND: menu;
  FONT-FAMILY: Arial, Helvetica, sans-serif;
  HEIGHT: 1px
}

```

```
.MainHeader
{
  FONT-SIZE: 14pt;
  COLOR: Darkblue;
  FONT-FAMILY: Arial, Helvetica, sans-serif
}

```

```
.DataCol
{

```



```

    FONT-SIZE: 8pt;
    FONT-FAMILY: Arial, Helvetica, sans-serif;
}

/*****
/** A HTML FILE THAT SHOWS AN EXAMPLE OF A WORKING      **/
/** DATA ISLAND. IT USES JAVASCRIPT IN ORDER TO PERFORM **/
/** CALCULATIONS.                                         **/
<!-- x-cart.html -->
<html>

<head><title>Scroll through the XML File - Using Data Islands</title></head>
<body bgcolor="beige">

<!-- Java Script required in order to ensure that a      -->
<!-- correct value was entered and to perform            -->
<!-- calculations (i.e. adding the total sales etc.      -->

<SCRIPT LANGUAGE="JavaScript">

function buyItem(newItem, newPrice, newQuantity)
{
    newItem = document.form1.SORT.value;
    newPrice = document.form1.PRICE.value;
    newQuantity = document.form1.COUNT.value;

    if (newQuantity <= 0)
    {
        rc = alert("Must have more than zero!.");
        return false;
    }
    if (newQuantity >= 0)
    {
        index = document.cookie.indexOf("MyCart");
        countbegin = (document.cookie.indexOf("=", index) + 1);
        countend = document.cookie.indexOf(";", index);
        if (countend == -1)
        {
            countend = document.cookie.length;
            window.location = "basket3.html";
        }
        document.cookie="MyCart="+document.cookie.substring(countbegin,
countend)+"["+newItem+", "+newPrice+"^"+newQuantity+"]";
    }
    return true;
}

//reset to null and 0

function resetShoppingCart()
{
    index = document.cookie.indexOf("MyCart");
    document.cookie="MyCart=";
}

</SCRIPT>

```

```

<!-- Java Script required to ensure that the correct -->
<!-- 'image' is displayed when the 'database' is -->
<!-- being scrolled through. -->

```

```

<SCRIPT LANGUAGE="JavaScript">
  if(document.images)
  {
    xover = new Array(9);
    xout = new Array(9);
    xover[1]=new Image;
    xout[1]=new Image;
    for(var n=2;n<=8;n++)
    {
      xover[n]=new Image;
      xout[n]=new Image;
    }
    for(var n=1;n<=8;n++) {

      xover[n].src="" + n + ".gif";
      xout[n].src="" + n + ".gif";
    }
  }
  function xOn(i)
  {
    if(document.images) document.images["x" + i].src=xover[i].src;
  }
  function xOff(i)
  {
    if(document.images) document.images["x" + i].src=xout[i].src;
  }
</script>

```

```

<center>
<table border=0 cellpadding=0 cellspacing=0 width=675>
<tr><td bgcolor="beige" width=675 height=25>&nbsp;</td></tr>
</table>
<table border=0 cellpadding=0 cellspacing=0 width=675>
<tr>

  <td bgcolor="beige" width=475>
  <center>
    <font face="arial" size=6 color=blue><b>Scroll through an XML File!</b></font>
  </center>
  <center>
    <font face="arial" color="#111111" size=2>Click on the grey button to order.</font>
  </center><P>

```

```

<div id="layer1" style="position:absolute; left:280; top:140;
width:120; height:80; background-color:beige; border: 3pt ridge wheat;">
  <center>
    <font face="arial" size=4 color=blue><b>ID Number</b></font>
  </center></div>
<div id="layer2" style="position:absolute; left:415; top:100;
width:145; height:160; background-color:beige; border: 3pt ridge wheat;">
  <center>
    <font face="arial" size=4 color=blue><b>Image</b></font>
  </center>
</div>

```

```
<!-- Embed the XML document that will be used to retrieve -->
<!-- the data from. -->
```

```
<XML id="myProducts" SRC="products1.xml"></XML>
```

```
<form name="form1">
  &nbsp;&nbsp;&nbsp;<font face="arial" size=4 color=blue><b>Book Title:</b></font>
  <P>&nbsp;&nbsp;&nbsp;<button ID="SORT" DATASRC="#myProducts" DATAFLD="SORT"
  onClick="buyItem(",",1);"></button><P>
  &nbsp;&nbsp;&nbsp;<font face="arial" size=4 color=blue><b>Price:</b>
  </font>&#x00a0;<span id="showPRICE" DATASRC="#myProducts"
  DATAFLD="PRICE"></span><input id="PRICE" type=hidden DATASRC="#myProducts"
  DATAFLD="PRICE"><P>
  &nbsp;&nbsp;&nbsp;<font face="arial" size=4 color=blue><b>Quantity:</b>
  </font><input id="COUNT" size=2 maxlength=2 value=1>
```

```
<div id="layer1a"
  style="position:absolute;left:305; top:170; width:70; height:30;
  background-color:whitesmoke; border: 4pt inset honeydew;"
  DATASRC="#myProducts" DATAFLD="IDNUM"></div><P>
<div id="layer2a"
  style="position:absolute;left:430; top:130; width:108; height:108;
  background-color:papayawhip; border: 4pt inset bisque;"
  DATAFORMATAS="HTML" DATASRC="#myProducts" DATAFLD="IMAGE">
</div><P>
```

```
<center><table border=0 cellpadding=0 cellspacing=0>
  <tr>
    <td><img name="x1" SRC="1.gif" alt="bottom"
    onclick="myProducts.recordset.MoveLast()"
    title="Last Record" onMouseOver=xOn("1") onMouseOut=xOff("1")></td>
    <td><img name="x2" SRC="2.gif" alt="next"
    onclick="if (! myProducts.recordset.EOF) myProducts.recordset.MoveNext();
    else myProducts.recordset.MoveFirst()" title="Next Record" onMouseOver=xOn("2")
    onMouseOut=xOff("2")></td>
    <td><img name="x3" SRC="3.gif" alt="prev"
    onclick="if (! myProducts.recordset.BOF) myProducts.recordset.MovePrevious();
    else myProducts.recordset.MoveLast()" title="Previous Record" onMouseOver=xOn("3")
    onMouseOut=xOff("3")></td>
    <td><img name="x4" SRC="4.gif" alt="top"
    onclick="myProducts.recordset.MoveFirst()"
    title="First Record" id="cmdFirst" onMouseOver=xOn("4") onMouseOut=xOff("4")></td>
  </tr>
</table></form></center>
```

```
<center>
  <font face="arial" color="#111111" size=2>Click arrows to move through database.</font>
</center>
  <td>
    <td bgcolor="beige" width=50>&nbsp;&nbsp;&nbsp;</td>
    <td bgcolor="beige" width=50>&nbsp;&nbsp;&nbsp;</td>
  </tr>
</table>
<table border=0 cellpadding=0 cellspacing=0 width=675><tr><td bgcolor="beige" width=675
height=40>&nbsp;&nbsp;&nbsp;</td>
```

```
<!-- Display all the details (the order) in a separate window -->
```

```
<FORM METHOD="GET" ACTION="basket3.html" TARGET="_blank">
  <INPUT TYPE="submit" VALUE="View Order">
```

```

</FORM>

<!-- Embed the SVG link -->

<EMBED SRC="x:\menu\HomeLink.svg" NAME="SVGEmbed" HEIGHT="33" WIDTH="97"
TYPE="image/svg+xml" PLUGINSOURCE="http://www.adobe.com/svg/viewer/install/">

<FORM ACTION="bogus_submit.html" METHOD="POST">
  <INPUT NAME="data" TYPE="hidden">
</FORM>
</tr>

</table>

</center>
</body>
</html>

```

```

//*****
/** THE XML FILE TO BE USED IN THE EXAMPLE ABOVE **/
<!-- products1.xml -->
<?xml version="1.0"?>
<PRODUCTS>
  <PRODUCT>
    <IDNUM>br3028</IDNUM>
    <SORT>XML Databases</SORT>
    <PRICE>23.87</PRICE>
    <IMAGE><![CDATA[<IMG SRC="13.jpg">]]></IMAGE>
  </PRODUCT>
  <PRODUCT>
    <IDNUM>br3029</IDNUM>
    <SORT>VoiceXML</SORT>
    <PRICE>28.99</PRICE>
    <IMAGE><![CDATA[<IMG SRC="16.jpg">]]></IMAGE>
  </PRODUCT>
  <PRODUCT>
    <IDNUM>br3030</IDNUM>
    <SORT>SMIL</SORT>
    <PRICE>19.25</PRICE>
    <IMAGE><![CDATA[<IMG SRC="11.jpg">]]></IMAGE>
  </PRODUCT>
  <PRODUCT>
    <IDNUM>br3031</IDNUM>
    <SORT>VML</SORT>
    <PRICE>19.03</PRICE>
    <IMAGE><![CDATA[<IMG SRC="12.jpg">]]></IMAGE>
  </PRODUCT>
  <PRODUCT>
    <IDNUM>br3032</IDNUM>
    <SORT>SVG</SORT>
    <PRICE>24.07</PRICE>
    <IMAGE><![CDATA[<IMG SRC="15.jpg">]]></IMAGE>
  </PRODUCT>
  <PRODUCT>
    <IDNUM>br3033</IDNUM>
    <SORT>MathML</SORT>
    <PRICE>24.50</PRICE>
    <IMAGE><![CDATA[<IMG SRC="11.jpg">]]></IMAGE>
  </PRODUCT>
</PRODUCTS>

```

```

<IDNUM>br3034</IDNUM>
<SORT>XML</SORT>
<PRICE>25.10</PRICE>
<IMAGE><![CDATA[<IMG SRC="14.jpg">]]></IMAGE>
</PRODUCT>
<PRODUCT>
  <IDNUM>br3035</IDNUM>
  <SORT>Java</SORT>
  <PRICE>27.16</PRICE>
  <IMAGE><![CDATA[<IMG SRC="12.jpg">]]></IMAGE>
</PRODUCT>
</PRODUCTS>

//*****
/** A HTML FILE THAT WILL DISPLAY THE PRODUCTS          **/
/** 'PURCHASED' ABOVE AND WILL DISPLAY THE TOTAL PRICE OF **/
/** THE PRODUCTS PURCHASED                             **/
<!-- basket3.html -->
<HTML>
<BODY>

<SCRIPT LANGUAGE="JavaScript">
function MakeTen(expr,decimal)
{
  var str="" + Math.round(eval(expr) * Math.pow(10,decimal))
  while (str.length <= decimal)
  {
    str = "0" + str
  }
  var decpoint = str.length - decimal
  return str.substring(0,decpoint) + "." + str.substring(decpoint,str.length);
}

function MakeTwo(expr)
{
  return "$" + MakeTen(expr,2)
}

function cartMe()
{
  index = document.cookie.indexOf("MyCart");
  countbegin = (document.cookie.indexOf("=", index) + 1);
  countend = document.cookie.indexOf(";", index);
  if (countend == -1)
  {
    countend = document.cookie.length;
  }
  fulllist = document.cookie.substring(countbegin, countend);
  totprice = 0;
  document.write('<TABLE ALIGN="CENTER" BORDER="1" CELLSPACING="2"
  CELLPADDING="2" BGCOLOR="navajowhite">');
  document.write('<TR><TD><B>Qty</B></TD><TD><B>Product</B></TD><TD><
  B>Price</B></TD><TD><B>Sub total</B></TD><TD>&nbsp;</TD></TR>');
  itemlist = 0;
  for (var i = 0; i <= fulllist.length; i++)
  {
    if (fulllist.substring(i,i+1) == '[')
    {
      itemstart = i+1;
    }
  }
}

```

```

else if (fulllist.substring(i,i+1) == ']')
{
    itemend = i;
    thequantity = fulllist.substring(itemstart, itemend);
    itemtotal = 0;
    itemtotal = (eval(theprice*thequantity));
    temptotal = itemtotal * 100;
    totprice = totprice + itemtotal;
    itemlist=itemlist+1;
    document.write('<TR><TD
ALIGN="CENTER">'+thequantity+'</TD><TD
ALIGN="LEFT">'+theitem+'</TD><TD
ALIGN="RIGHT">$'+theprice+'</TD><TD
ALIGN="RIGHT">'+self.MakeTwo(itemtotal)+'</TD><TD><A
href="javascript:removeItem('+itemlist+')">Remove</A></TD></TR>');
}
else if (fulllist.substring(i,i+1) == ',')
{
    theitem = fulllist.substring(itemstart, i);
    itemstart = i+1;
}
else if (fulllist.substring(i,i+1) == '^')
{
    theprice = fulllist.substring(itemstart, i);
    itemstart = i+1;
}
else if (fulllist.substring(i,i+1) == '~')
{
    theoption = fulllist.substring(itemstart, i);
    itemstart = i+1;
}
}
document.write('<TR><TD ALIGN="RIGHT"
colspan=3><B>Total</B></TD><TD
ALIGN="RIGHT">'+self.MakeTwo(totprice)+'</TD><TD>&nbsp;</TD></TR>');
document.write('</TABLE>');
}

```

```

function removeItem(itemno)
{
    newItemList = null;
    itemlist = 0;
    for (var i = 0; i <= fulllist.length; i++)
    {
        if (fulllist.substring(i,i+1) == '[')
        {
            itemstart = i+1;
        }
        else if (fulllist.substring(i,i+1) == ']')
        {
            itemend = i;
            theitem = fulllist.substring(itemstart, itemend);
            itemlist=itemlist+1;
            if (itemlist != itemno)
            {
                newItemList = newItemList+'['+fulllist.substring(itemstart,
itemend)+']';
            }
        }
    }
}

```

```

    }

    index = document.cookie.indexOf("MyCart");
    document.cookie="MyCart="+newItemList;
    window.location = "basket3.html";
}

function clearCart()
{
    if (confirm('Are you sure you wish to clear the cart?'))
    {
        index = document.cookie.indexOf("MyCart");
        document.cookie="MyCart=";
        window.location = "basket3.html";
    }
}
</SCRIPT>

<TABLE ALIGN ="CENTER" WIDTH="500" BORDERCOLOR="burlywood" BGCOLOR="beige"
BORDER="2" CELLPADDING="0" CELLSPACING="0" HEIGHT="350">
<TR><TD>
<CENTER><B><FONT SIZE=6 COLOR="blue">Items Ordered</FONT></B></CENTER>
<P>&nbsp;</P>
<SCRIPT LANGUAGE="JavaScript">
</SCRIPT>

<TABLE ALIGN ="CENTER" BORDER=0 CELLSPACING=2 CELLPADDING=3>
<TR VALIGN=Top>
<TD><FORM><INPUT TYPE="BUTTON" NAME="clear" VALUE="Clear Cart"
ONCLICK="clearCart()"></TD>
</TR></TABLE>

</TD></TR>
</TABLE>
</BODY>
</HTML>

/*****
/** HTML PAGE THAT HAS AN SVG FILE EMBEDDED IN IT **/
/** AND REFERS TO TWO JAVASCRIPT FILES THAT WILL **/
/** ENABLE THE USER TO 'DRAG' AND 'DROP' SVG ICONS **/
<!-- book.html -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>By Nigel McKelvey 2002</TITLE>

<!-- Call the Java Scripts used to -->
<!-- drag and drop the icons -->

<script src="drag.js"></script>
<script src="forms.js"></script>
</HEAD>
<BODY BGCOLOR="beige">
<center>

<!-- Call the SVG File -->

<EMBED SRC="book.svg" NAME="SVGEmbed" HEIGHT="360" WIDTH="1000"
TYPE="image/svg+xml" PLUGINSOURCE="http://www.adobe.com/svg/viewer/install/">

```



```

<FORM ACTION="bogus_submit.html" METHOD="POST">
  <INPUT NAME="data" TYPE="hidden">
</FORM>

<!-- link to the home page -->
<EMBED SRC="x:\menu\HomeLink.svg" NAME="SVGEmbed" HEIGHT="33" WIDTH="97"
TYPE="image/svg+xml" PLUGINSOURCE="http://www.adobe.com/svg/viewer/install/">

<FORM ACTION="bogus_submit.html" METHOD="POST">
  <INPUT NAME="data" TYPE="hidden">
</FORM>
</center>

</BODY>
</HTML>

//*****
//forms.js
// Author: Nigel McKelvey
// Date: 2002

// This set of functions is designed to parse a <g> containing a <text> node and generate
// an HTML <form>.
// The <text> is expected to be composed of sets of 3 <tspan>s.
// The first <tspan> contains the label, the second <tspan> a separator,
// and the third <tspan> the current value of the field. e.g.:

var formWindow = null;

//-----

// Function to create a window with specified dimensions

function BuildForm(evt)
{
  var target = get_target(evt);

  var objectText = get_objectText(target);
  if (objectText != null)
  {
    var myForm = MakeForm(objectText);

    formWindow =
    window.open("", "formWindow", "HEIGHT=400,WIDTH=400,scrollbars,resizable");
    formWindow.document.open();
    formWindow.document.write(myForm);
    formWindow.document.close();
  }
}

//-----

function MakeForm (node)
{
  var formText = null;

```

```

var textChildren = node.getElementsByTagName("text");

// Get a list of all the children of the node

if (textChildren)
{
    var i = 0;
    // Find the first text node with <tspan> children.
    do
    {
        textChild = textChildren.item(i);
        tspanChildren = textChild.getElementsByTagName("tspan");
        i++;
    } while ((textChild != null) && (tspanChildren == null));

    if (tspanChildren == null)
    {
        // If no <tspan>s are found then stop
        return null;
    }

    // if tspans are found then start to make the form
    formText = "<FORM>\n";

    for (var x = 0; x < tspanChildren.length; x+=3 )
    {
        // the item may be empty and thus have no children.
        var itemValue = "";
        if (tspanChildren.item(x+2).firstChild != null)
        {
            itemValue = tspanChildren.item(x+2).firstChild.nodeValue;
        }
        formText += "<P><STRONG>" +
            tspanChildren.item(x).firstChild.nodeValue +
            tspanChildren.item(x+1).firstChild.nodeValue + "</STRONG><BR>\n";
        formText += "<INPUT TYPE='text' SIZE='40' NAME='" +
            tspanChildren.item(x).firstChild.nodeValue + "' VALUE='" + itemValue +
            "'></P>\n";
    }

    formText += "<INPUT TYPE='button' VALUE='Update'
onClick='opener.ProcessForm(this.form)'>";
    formText += "</FORM>";

}
return formText;
}

//-----

// Insert the new values back into the original SVG DOM.
// Refresh the text info display.

function ProcessForm(theForm)
{
    formWindow.close();

    for (x = 0; x < theForm.elements.length-1; x++)
    // The last element of the form is the 'Update'

```

```

//button
{
    var svgdoc = currObject.getOwnerDocument();
    //currObject is a global variable that tells which
    //object the text in the right panel goes with.

    var currValue = theForm.elements[x].value;
    var textChildren = currObject.getElementsByTagName("text");

    // Get a list of all the children of this node.

    textChild = textChildren.item(0);
    tspanChildren = textChild.getElementsByTagName("tspan");

    if (tspanChildren == null)
    {
        return null;
    }

    if (tspanChildren.item((x*3)+2).firstChild != null)
    {
        tspanChildren.item((x*3)+2).firstChild.nodeValue = currValue;
    }
    else
    {
        //Add a text node

        newNode = svgdoc.createTextNode(currValue);
        tspanChildren.item((x*3)+2).appendChild(newNode);
    }
}

var textInfo = svgdoc.getElementById ('textInfo');
var textInfoText = get_objectText(textInfo);
if (textInfoText != null)
{
    textInfo.removeChild(textInfoText);
}
currObject = null;
}
return true;
}

//-----

// Submit the modified data back to the server.

function SubmitData(evt)
{
    var text = "";

    var contents = getSVGDocument(get_target(evt)).getElementById("contents");

    var groups = contents.getElementsByTagName("g");

```

```

if (groups.length > 0)
{
    text += "<?xml version='1.0' encoding='UTF-8'>\n";
    text += "<bookdata xmlns:xsi='http://www.w3.org/1999/XMLSchema-
instance' xsi:noNamespaceSchemaLocation='bookdata.xsd'>\n";
}

for (x=0; x < groups.length; x++)
{
    // for each group look for an 'objectText' group
    objectText = get_objectText (groups.item(x));

    if (objectText != null)
    {

        var textChildren = objectText.getElementsByTagName("text");
        // Get a list of all the children of this node
        textChild = textChildren.item(0);
        tspanChildren = textChild.getElementsByTagName("tspan");

        if (tspanChildren == null)
        {

            return null;

        }

        text += "<" + groups.item(x).getAttribute("id") + ">\n";
        for (y=0; y < tspanChildren.length; y+=3)
        {
            text += "<" +
            tspanChildren.item(y+2).getAttribute("desc") +
            ">";
            if (tspanChildren.item(y+2).firstChild != null)
            {
                text +=
                tspanChildren.item(y+2).firstChild.node
                Value;
            }
            text += "</" +
            tspanChildren.item(y+2).getAttribute("d
            esc") + ">\n";
        }
        text += "</" + groups.item(x).getAttribute("id") + ">\n";
    }
}
if (groups.length > 0)
{
    text += "</bookdata>";
}
document.forms[0].elements[0].value = text;
document.forms[0].submit();
}

```

```

function RemoveObject(evt)
{
    // currObject points to the "objectText" group of the object,
    // find the parent and remove the parent.
    var parent = currObject.parentNode;
}

```

```

var grandParent = parent.parentNode;
grandParent.removeChild(parent);
toggleVis2(evt);
}

```

```

//-----

```

```

//Pop up a window with the icon's data and confirm that the user
//wants to delete this item. Then delete it, if yes.

```

```

function ConfirmDelete (evt)
{
    var formText = null;

    var target = get_target(evt);

    var node = get_objectText(target);

    if (node == null)
    {
        return;
    }

    var textChildren = node.getElementsByTagName("text");
    if (textChildren)
    {
        var i = 0;
        // Find the first text node with <tspan> children.
        do
        {
            textChild = textChildren.item(i);
            tspanChildren = textChild.getElementsByTagName("tspan");
            i++;
        } while ((textChild != null) && (tspanChildren == null));

        if (tspanChildren == null)
        {
            return null;
        }

        formText="Are you sure you want to delete the following book?\n";
        for (var x = 0; x < tspanChildren.length; x+=3 )
        {
            var myNodeValue = "";
            if (tspanChildren.item(x+2).firstChild != null)
            {
                myNodeValue =
                tspanChildren.item(x+2).firstChild.nodeValue;
            }

            //ask the user the exact details

            formText += tspanChildren.item(x).firstChild.nodeValue +
            tspanChildren.item(x+1).firstChild.nodeValue;
            formText += myNodeValue + "\n";
        }

        if (confirm(formText))

```

```

        }
        RemoveObject(evt);
    }
}

//-----

//*****
//drag.js
// Author: Nigel McKelvey
// Date: 2002

var NWlatBound = 39.171111;
    var SElatBound = 39.165833;
    var NWlongBound = -123.224444;
    var SElongBound = -123.215278;
    var mapWidth = 800;
    var mapHeight = 600;

    // Current item descriptive text being displayed.
    var currText = null;

    // Current object within "content" group whose values are being displayed in
    righthand panel. This is used to get edited values back to the original object.

    var currObject = null;

    var dragger = null;
    var origTransform = "";
    var origStyle = "";
    var origX;
    var origY;
    var oldTranslateX;
    var oldTranslateY;
    var translateRegExp=/translate\(([+-]?\d+)(\s*(\s,)\s*)([+-]?\d+)\)\s*/

    function DoNothing()
    {
    }

    function DoOnLoad(evt)
    {
    }

    function DoOnMouseOver(evt)
    {
    }

    function DoOnMouseOut(evt)
    {
    }

    function DoOnMouseMove(evt)
    {
        if( dragger != null )
        {

```

```

        SetTranslateText(evt);
    }

    // Propagate the event to other handlers.
    return true;
}

function DoOnMouseDown(evt)
{
    var onObject = get_target(evt);
    if(onObject.getAttribute("drag") != "false")
    {

        if (onObject.getAttribute("clone") == "true")
        {
            onObject = clone_me( evt );
        }

        dragger = onObject;

        origTransform = dragger.getAttribute("transform");
        origStyle = dragger.getAttribute("style");

        if (origTransform == null)
            origTransform = "";
        else
            origTransform = new String(origTransform);

        origX = evt.getClientX();
        origY = evt.getClientY();
        oldTranslateX = 0;
        oldTranslateY = 0;

        if (origTransform.length != 0)
        {
            var result = origTransform.match(translateRegExp);
            if (result == null || result.index == -1)
            {
                alert("The regular expression had a problem
                finding the translate at the end of\'" +
                origTransform + "\'");
                oldTranslateX = 0;
                oldTranslateY = 0;
            }
            else
            {
                oldTranslateX = parseFloat(result[1]);
                oldTranslateY = parseFloat(result[3]);
                origTransform = origTransform.substr(0,
                result.index);
            }
        }

        if (onObject.getAttribute("id") != "shapeBar")
        {
            SetTranslateText(evt);

            var el =
            getSVGDocument(onObject).getElementById("coordText
            ");
        }
    }
}

```



```

        el.getStyle().setProperty("visibility", "visible");
    }
}

// Propagate the event to other handlers.
return true;
}

function DoOnMouseUp(evt)
{
    if( dragger != null )
    {
        if (dragger.getAttribute("id") != "shapeBar")
        {
            dragger.setAttribute("style", origStyle);

            if (dragger.getAttribute("clone") == "false")
            // must be an icon on the map.
            {
                // Update the Lat & Long elements.
                UpdateLatLong(evt, dragger);
                toggleVis2(evt);
            }

            origTransform = "";
            origX = 0;
            origY = 0;
            oldTranslateX = 0;
            oldTranslateY = 0;

            var el = getSVGDocument(dragger).getElementById("coordText");
            el.getStyle().setProperty("visibility", "hidden");

            dragger = null;
        }
    }
    // Propagate the event to other handlers.
    return true;
}

function get_objectText (target)
{
    var children = target.getElementsByTagName("g");
    // Get a list of all the children of this node.
    if (children)
    {
        var i=0;
        var currChild = children.item(i);
        // Pick up the first child node.

        while (currChild)
        {
            if (currChild.getNodeType() == 1)
            {
                // if the current node is an element node.
                if (currChild.getAttribute("desc") ==
                    "objectText")
                {

```

```

        return currChild;
    }
    }
    currChild = children.item(++i);
}
return null;
}
return null;
}

function FindXCoord (evt, target)
{
    var newX = oldTranslateX + (evt.getClientX() - origX);
    var xOffset = target.getAttribute('width');
    var finalX = Math.round(parseInt(newX) + (parseInt(xOffset) / 2));

    return finalX;
}

function FindYCoord(evt, target)
{
    var newY = oldTranslateY + (evt.getClientY() - origY);
    var yOffset = target.getAttribute('height');
    var finalY = Math.round(parseInt(newY) + (parseInt(yOffset) / 2));

    return finalY;
}

function UpdateLatLong(evt, target)
{
    var newNode = null;
    var objectText = get_objectText(target);
    // take the <text> node.
    var textChildren = objectText.getElementsByTagName("text");
    // Get a list of all the children of this node.
    var textChild = textChildren.item(0);
    var tspanChildren = textChild.getElementsByTagName("tspan");

    var currLat = NWlatBound - ((FindYCoord(evt, target)/mapHeight) *
    (NWlatBound - SElatBound));
    var currLong = NWlongBound - ((FindXCoord(evt, target)/mapWidth) *
    (NWlongBound - SElongBound));
    var latString = FormatNumber(currLat, 6);
    var longString = FormatNumber(currLong, 6);

    for (var x = 0; x < tspanChildren.getLength(); x+=3 )
    {
        if (tspanChildren.item(x+2).getAttribute("desc") == "Latitude")
        {
            if (tspanChildren.item(x+2).firstChild != null)
            {
                tspanChildren.item(x+2).firstChild.nodeValue =
                latString;
            }
            else
            {
                // add a text node to the <tspan>
                var svgdoc = target.getOwnerDocument();
                newNode = svgdoc.createTextNode(latString);
            }
        }
    }
}

```

```

        tspanChildren.item(x+2).appendChild(newNode);
    }
}

if (tspanChildren.item(x+2).getAttribute("desc") == "Longitude")
{
    if (tspanChildren.item(x+2).firstChild != null)
    {
        tspanChildren.item(x+2).firstChild.nodeValue =
            longString;
    }
    else
    {
        // add a text node to the <tspan>
        var svgdoc = target.getOwnerDocument();
        newNode = svgdoc.createTextNode(longString);
        tspanChildren.item(x+2).appendChild(newNode);
    }
}
}
}

function toggleVis2(evt)
{
    // Retrieve node of object that was clicked on.
    var target = get_target(evt);
    var svgdoc = target.getOwnerDocument();

    // Retrieve the node for "textInfo" group.
    var textInfo = svgdoc.getElementById ('textInfo');
    var textInfoText = get_objectText(textInfo);

    if (textInfoText != null)
    {
        textInfo.removeChild(textInfoText);
    }

    var svgobj = get_objectText(target);

    if (svgobj != null)
    {
        var newnode = svgobj.cloneNode(true);
        // Make a copy of the node
        // include all its children.

        textInfo.appendChild(newnode);
        newnode.getStyle().setProperty ('visibility', 'visible');
        currObject = svgobj;
        // Set global var to keep track of which object this display
        is associated with.
    }
    // Propagate the event to other handlers.
    return true;
}

// Change the visibility of the object
function toggleVis (evt)

```

```

    {
        var target = get_target(evt);
        var svgobj = get_objectText(target);

        if (svgobj != null)
        {
            var visStyle = "";
            var svgstyle = svgobj.getStyle();
            visStyle = svgobj.getAttribute("style");
            if (visStyle.match("hidden"))
            {
                if (currText != null)
                {
                    currText.getStyle().setProperty ('visibility',
                        'hidden');
                }
                currText = svgobj;

                svgstyle.setProperty ('visibility', 'visible');
            }
            else
            {
                svgstyle.setProperty ('visibility', 'hidden');
                currText = null;
            }
        }
        // Propagate the event to other handlers.
        return true;
    }

function SetTranslateText( evt )
{
    var target = get_target(evt);
    var newX = oldTranslateX + (evt.getClientX() - origX);
    var newY = oldTranslateY + (evt.getClientY() - origY);

    var finalX = FindXCoord(evt, target);
    var finalY = FindYCoord(evt, target);

    var transform = origTransform + "translate(" + newX + " " + newY
        + ")";

    var text = getSVGDocument(target).getElementById("coordText");

    var currLat = NWlatBound - ((finalY/mapHeight) * (NWlatBound
        - SElatBound));
    var currLong = NWlongBound - ((finalX/mapWidth) *
        (NWlongBound - SElongBound));
    var latString = FormatNumber(currLat, 6);
    var longString = FormatNumber(currLong, 6);

    text.getFirstChild().setData("(" + latString + ", " + longString +
        ")\r\n(" + finalX + ", " + finalY + ")");

    dragger.setAttribute("transform", transform );
}

function getSVGDocument(node)
{

```

lyit

Institiúid Teicneolaíochta Leitir Ceannainn
Letterkenny Institute of Technology

```

        // given any node of the tree, will obtain the SVGDocument node.
        if( node.getNodeType() != 9 )
        // if not DOCUMENT_NODE
            return node.getOwnerDocument();
        else
            return node;
    }

//returns the SVG object
function get_target (evt)
{
    var target = evt.getTarget();
    while (target && !target.getAttribute('id'))
        target = target.parentNode();
    return target;
}

//creates a duplicate of the icon that the user clicks on
function clone_me (evt)
{
    // Retrieve node of object that was clicked on.
    var target = get_target(evt);

    var svgdoc = target.getOwnerDocument();

    // Find values for the new elements x, y, and fill attributes
    // and properties.
    var x = evt.getClientX();
    var y = evt.getClientY();

    // Clone the object and set its attributes and properties.
    var newnode = target.cloneNode(true);
    newnode.setAttribute ('drag', 'true');
    newnode.setAttribute ('clone', 'false');
    newnode.getStyle().setProperty('opacity', '0.7');

    var xOffset = newnode.getAttribute('width');
    var yOffset = newnode.getAttribute('height');

    var finalX = Math.round(parseInt(x) - (parseInt(xOffset) / 2));
    var finalY = Math.round(parseInt(y) - (parseInt(yOffset) / 2));

    var transform = "translate(" + finalX + " " + finalY + ")";
    newnode.setAttribute("transform", transform );

    // Retrieve the node for "contents" group.
    var contents = svgdoc.getElementById ('contents');

    // Insert the cloned object into the contents group node.
    newnode = contents.appendChild (newnode);

    return newnode;
}

function FormatNumber(expr, decplaces)
{
    var str = "" + Math.round(eval(expr) * Math.pow(10, decplaces));
    while (str.length <= decplaces)
    {

```

```

        str += "0";
    }

    var decpoint = str.length - decplaces;

    return str.substr(0,decpoint) + "." + str.substr(decpoint,str.length);
}

function PressButton(evt, pressed)
{
    target = evt.getTarget();
    if (pressed == true)
    {
        target.getStyle().setProperty('filter', 'none');
    }
    else
    {
        target.getStyle().setProperty('filter',
        'url(#closeDropShadow)');
    }
}
}

/*****
/** SVG PAGE THAT CREATES THE BOOK ICONS MENTIONED ABOVE      **/
/** AND PROVIDES AN INTERFACE FOR THE USER TO                  **/
/** MANIPULATE THE DATA CONTAINED WITHIN THESE ICONS.       **/
<!-- book.svg -->
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg SYSTEM "http://www.w3.org/TR/2000/03/WD-SVG-20000303/DTD/svg-
20000303-stylable.dtd">
<svg enableZoomAndPanControls="false" height="600" width="600">

<!-- ***** -->
<!-- Nigel McKelvey -->
<!-- 2003 -->
<!-- This program will allow the user to add/delete -->
<!-- book icons using XML as a temporary database -->
<!-- ***** -->

<defs>
  <filter height="350%" width="250%" y="-70%" x="-70%" filterUnits="objectBoundingBox"
    id="closeDropShadow">
    <feGaussianBlur result="blur" stdDeviation="2" in="SourceAlpha"/>
    <feOffset result="offsetGraphic" dy="-2" dx="-2" in="SourceGraphic"/>
    <feMerge>
      <feMergeNode in="blur"/>
      <feMergeNode in="offsetGraphic"/>
    </feMerge>
  </filter>

  <g id="house_def" style="stroke:#000000;stroke-miterlimit:4;">

<!-- Create the icon -->

  <circle stroke="black" fill="white" cy="100px" cx="90px" r="30px"/>
  <text y="105px" x="66px" style="font-weight: bold;
    font-family: times; font-size: 16pt">Book</text>

```



```

</g>

<!-- Create the buttons -->

<g id="editButton_def">
  <rect style="fill:#AFFFEE; stroke:#FF2300; stroke-width:3;" ry="4" rx="4" height="20"
    width="60"/>

  <text startOffset="0" style="font-family:times; font-size:12pt; fill:black;" y="1em"
    x="5">Edit</text>
</g>
<g id="deleteButton_def">
  <rect style="fill:#AFFFEE; stroke:#FF2300; stroke-width:3;" ry="4" rx="4" height="20"
    width="60"/>
  <text startOffset="0" style="font-family:times; font-size:12pt; fill:black;" y="1em"
    x="5">Delete</text>
</g>

</defs>

<g onmousemove="DoOnMouseMove(evt)"
  onmouseup="DoOnMouseUp(evt)"
  onmousedown="DoOnMouseDown(evt)"
  onmouseout="DoOnMouseOut(evt)"
  onmouseover="DoOnMouseOver(evt)"
  drag="false" id="contents">

  <rect style="fill:beige;" height="900" width="650" y="-50" x="-50"/>
  <g style="opacity:0.7;" drag="true" height="25" width="25" clone="false" id="house"
    transform="translate(410 212)">

  <use style="fill:grey" xlink:href="#house_def"/>
  <g style="font-family:Verdana; font-size:10pt; fill:black; visibility:hidden;"
    desc="objectText" transform="translate(0 25)">

  <text startOffset="0">

  <!-- Assign values to each icon -->

  <tspan startOffset="2" dy="1.25em" x="5">Title</tspan>
  <tspan startOffset="0">: </tspan>
  <tspan startOffset="0" dy="1.0em" x="10" desc="Title">XML Databases</tspan>
  <tspan startOffset="0" dy="1.25em" x="5">ISBN</tspan>
  <tspan startOffset="0">: </tspan>
  <tspan startOffset="0" dy="1.0em" x="10" desc="ISBN">13334265</tspan>
  <tspan startOffset="0" dy="1.25em" x="5">Author</tspan>
  <tspan startOffset="0">: </tspan>
  <tspan startOffset="0" dy="1.0em" x="10" desc="Author">Nigel A McKelvey</tspan>
  <tspan startOffset="0" dy="1.25em" x="5">Date</tspan>
  <tspan startOffset="0">: </tspan>
  <tspan startOffset="0" dy="1.0em" x="10" desc="Date">08-10-02</tspan>
  <tspan startOffset="0" dy="1.25em" x="5">Price</tspan>
  <tspan startOffset="0">: </tspan>
  <tspan startOffset="0" dy="1.0em" x="10" desc="Price">89.99</tspan>
  <tspan startOffset="0" dy="1.25em" x="5">Web Site</tspan>

```

```

    <tspan startOffset="0">: </tspan>
    <tspan startOffset="0" dy="1.0em" x="10" desc="WebSite">www.nigelsbook.com</tspan>

  </text>
</g>
</g>
<g style="opacity:0.7;" drag="true" height="25" width="25" clone="false" id="house"
transform="translate(366 173)">
<use style="fill:grey" xlink:href="#house_def"/>
<g style="font-family:Verdana; font-size:10pt; fill:black; visibility:hidden;"
desc="objectText" transform="translate(0 25)">
  <text startOffset="0">

    <tspan startOffset="0" dy="1.25em" x="5">Title</tspan>
    <tspan startOffset="0">: </tspan>
    <tspan startOffset="0" dy="1.0em" x="10" desc="Title">Java Enterprises</tspan>
    <tspan startOffset="0" dy="1.25em" x="5">ISBN</tspan>
    <tspan startOffset="0">: </tspan>
    <tspan startOffset="0" dy="1.0em" x="10" desc="ISBN">222311211</tspan>
    <tspan startOffset="0" dy="1.25em" x="5">Author</tspan>
    <tspan startOffset="0">: </tspan>
    <tspan startOffset="0" dy="1.0em" x="10" desc="Author">Eilish S. Scott</tspan>
    <tspan startOffset="0" dy="1.25em" x="5">Date</tspan>
    <tspan startOffset="0">: </tspan>
    <tspan startOffset="0" dy="1.0em" x="10" desc="Date">01-01-01</tspan>
    <tspan startOffset="0" dy="1.25em" x="5">Price</tspan>
    <tspan startOffset="0">: </tspan>
    <tspan startOffset="0" dy="1.0em" x="10" desc="Price">40.00</tspan>
    <tspan startOffset="0" dy="1.25em" x="5">Web Site</tspan>
    <tspan startOffset="0">: </tspan>
    <tspan startOffset="0" dy="1.0em" x="10" desc="WebSite">www.eilishssite.ie</tspan>

  </text>
</g>
</g>
<g style="opacity:0.7;" drag="true" height="25" width="25" clone="false" id="house"
transform="translate(369 214)">
<use style="fill:grey" xlink:href="#house_def"/>
<g style="font-family:Verdana; font-size:10pt; fill:black; visibility:hidden;"
desc="objectText" transform="translate(0 25)">
  <text startOffset="0">

    <tspan startOffset="0" dy="1.25em" x="5">Title</tspan>
    <tspan startOffset="0">: </tspan>
    <tspan startOffset="0" dy="1.0em" x="10" desc="Title">Garbage Collection</tspan>
    <tspan startOffset="0" dy="1.25em" x="5">ISBN</tspan>
    <tspan startOffset="0">: </tspan>
    <tspan startOffset="0" dy="1.0em" x="10" desc="ISBN">000988787</tspan>
    <tspan startOffset="0" dy="1.25em" x="5">Author</tspan>
    <tspan startOffset="0">: </tspan>
    <tspan startOffset="0" dy="1.0em" x="10" desc="Author">Ruth Lennon</tspan>
    <tspan startOffset="0" dy="1.25em" x="5">Date</tspan>
    <tspan startOffset="0">: </tspan>
    <tspan startOffset="0" dy="1.0em" x="10" desc="Date">11-09-00</tspan>
    <tspan startOffset="0" dy="1.25em" x="5">Price</tspan>
    <tspan startOffset="0">: </tspan>
    <tspan startOffset="0" dy="1.0em" x="10" desc="Purchase_Price">56.99</tspan>
    <tspan startOffset="0" dy="1.25em" x="5">Web Site</tspan>
    <tspan startOffset="0">: </tspan>

  </text>
</g>
</g>

```

```

<tspan startOffset="0" dy="1.0em" x="10" desc="WebSite">www.lennon.ie</tspan>
</text>
</g>
</g>
<g style="opacity:0.7;" drag="true" height="25" width="25" clone="false" id="house"
transform="translate(389 118)">
  <use style="fill:grey" xlink:href="#house_def"/>
  <g style="font-family:Verdana; font-size:10pt; fill:black; visibility:hidden;" desc="objectText"
transform="translate(0 25)"><text startOffset="0">

  <tspan startOffset="0" dy="1.25em" x="5">Title</tspan>
  <tspan startOffset="0">: </tspan>
  <tspan startOffset="0" dy="1.0em" x="10" desc="Title">Building B2B
  Applicataions</tspan>
  <tspan startOffset="0" dy="1.25em" x="5">ISBN</tspan>
  <tspan startOffset="0">: </tspan>
  <tspan startOffset="0" dy="1.0em" x="10" desc="ISBN">01018988765</tspan>
  <tspan startOffset="0" dy="1.25em" x="5">Author</tspan>
  <tspan startOffset="0">: </tspan>
  <tspan startOffset="0" dy="1.0em" x="10" desc="Author">F. Scott Fitzgearld</tspan>
  <tspan startOffset="0" dy="1.25em" x="5">Date</tspan>
  <tspan startOffset="0">: </tspan>
  <tspan startOffset="0" dy="1.0em" x="10" desc="Date">23-05-97</tspan>
  <tspan startOffset="0" dy="1.25em" x="5">Price</tspan>
  <tspan startOffset="0">: </tspan>
  <tspan startOffset="0" dy="1.0em" x="10" desc="Price">39.99</tspan>
  <tspan startOffset="0" dy="1.25em" x="5">Web Site</tspan>
  <tspan startOffset="0">: </tspan>
  <tspan startOffset="0" dy="1.0em" x="10" desc="WebSite">www.fscott.org</tspan>

  </text>
</g>
</g>
</g>
<g transform="translate(805 5)" drag="false" id="textInfo">
  <rect style="fill:#FFFFFFE; stroke:#111122; stroke-width:2;" height="300" width="190"
ry="4" rx="4" y="0" x="0"/>
  <g onmousedown="PressButton(evt, true)" onmouseout="PressButton(evt, false)">
    <use style="filter:url(#closeDropShadow)" xlink:href="#editButton_def"
transform="translate(5 5)"
onmouseup="PressButton(evt, false); BuildForm(evt)"/>
    <use style="filter:url(#closeDropShadow)" xlink:href="#deleteButton_def"
transform="translate(129 5)"

  <!-- call the ConfirmDelete function -->

  onmouseup="PressButton(evt, false); ConfirmDelete(evt)"/>
</g>
</g>
<g id="shapeBar" transform="translate(540 20)" onmouseup="DoOnMouseUp( evt )"
onmousedown="DoOnMouseDown( evt )" onmousemove="DoOnMouseMove( evt )">

  <text x="5" y="40" id="coordText" drag="true" style="visibility: hidden; fill: beige; font-family:
Serif; font-size: 14">Hello!</text>

  <!-- Allow the user to create a new icon -->

```

```

<g id="house" clone="true" transform="translate(65 5)" width="25" height="25">
  <use xlink:href="#house_def" style="fill:grey"/>
  <g desc="objectText" transform="translate(0 25)" style="font-family:Verdana; font-size:10pt;
  fill:black; visibility:hidden;">
    <text startOffset="0">

      <tspan startOffset="0" dy="1.25em" x="5">Title</tspan>
      <tspan startOffset="0">: </tspan>
      <tspan startOffset="0" dy="1.0em" x="10" desc="Title"></tspan>
      <tspan startOffset="0" dy="1.25em" x="5">ISBN</tspan>
      <tspan startOffset="0">: </tspan>
      <tspan startOffset="0" dy="1.0em" x="10" desc="ISBN"></tspan>
      <tspan startOffset="0" dy="1.25em" x="5">Author</tspan>
      <tspan startOffset="0">: </tspan>
      <tspan startOffset="0" dy="1.0em" x="10" desc="Author"></tspan>
      <tspan startOffset="0" dy="1.25em" x="5">Date</tspan>
      <tspan startOffset="0">: </tspan>
      <tspan startOffset="0" dy="1.0em" x="10" desc="Date"></tspan>
      <tspan startOffset="0" dy="1.25em" x="5">Price</tspan>
      <tspan startOffset="0">: </tspan>
      <tspan startOffset="0" dy="1.0em" x="10" desc="Price"></tspan>
      <tspan startOffset="0" dy="1.25em" x="5">Web Site</tspan>
      <tspan startOffset="0">: </tspan>
      <tspan startOffset="0" dy="1.0em" x="10" desc="WebSite"/>

    </text>
  </g>
</g>
</svg>

/**
** A APPLET THAT CONTAINS A REFERENCE TO A JAVA FILE THAT
** CAN ACCESS DATA STORED IN AN XML FILE.
**

<!-- Author: Nigel McKelvey -->
<!-- 2002 -->
<!-- Applet4Search.html -->

<HTML>
  <HEAD>
    <TITLE>LYIT Book Shop</TITLE>
  </HEAD>
  <BODY bgcolor="beige" text="sienna">

    <p><h4 align="center">Search through the Temporary DB for a list of Books.</p>
    <p>Type the Author's surname into the box and click 'Search'</P    </h4>

    <!-- Call the Java class file and specify the XML file -->

      <CENTER>
        <APPLET CODE=XMLSearch2 WIDTH=300 HEIGHT=300>
          <PARAM NAME="url" VALUE="Book.xml"></APPLET>
      </CENTER>

    <br>

    <!-- Embed the SVG link -->

    <center>

```

```
<EMBED SRC="x:\menu\HomeLink.svg" NAME="SVGEmbed" HEIGHT="33"
WIDTH="97" TYPE="image/svg+xml"
PLUGINSPPAGE="http://www.adobe.com/svg/viewer/install/">
```

```
<FORM ACTION="bogus_submit.html" METHOD="POST">
  <INPUT NAME="data" TYPE="hidden">
</FORM>
</center>
```

```
<!-- Allow the user to print the contents of the page -->
```

```
<xml>
<MSHelp:Keyword Index="A" Term="print"/>
</xml>
```

Print Page

```
<input type=button value="Print" onclick="window.print()">
```

```
</BODY>
</HTML>
```

```
/**
*****
** THE JAVA FILE REFERRED RO ABOVE **
XMLSearch2.java
*****
*/
Author: Nigel McKelvey */
Date: 2002 */
This program will search throuh an XML file */
and retrieve any necessary information. */
The XML doucment is acting like a 'database' itself. */
*****
//-----
//libraries
import com.ms.xml.om.*;
import com.ms.xml.parser.*;
import com.ms.xml.util.*;
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
import java.util.*;
//-----
public class XMLSearch2 extends Applet
implements ActionListener
{
//declare variables required
private Document doc;
private URL xmlURL;
private TextArea textArea;
private TextField target;
private Button search;
private int numtargets;
public void init ()
```

```

    {
        //create the buttons, text area etc.

        Panel p = new Panel ();
        target = new TextField(20);
        search = new Button("Search");
        search.addActionListener(this);
        p.add (target);
        p.add (search);

        setLayout(new BorderLayout());
        textArea = new TextArea();
        add("Center", textArea);
        add("North",p);

        doc = new Document();

        //retrieve the name of the XML document

        String xmldoc = getParameter("url");
        try
        {
            xmlURL = new URL (getDocumentBase(), xmldoc);
        }
        catch (MalformedURLException e)
        {
            e.printStackTrace();
        }
        try
        {
            doc.load(xmlURL);
        }
        catch (ParseException e)
        {
            doc.reportError (e,System.out);
        }
    }

//-----

    public void actionPerformed (ActionEvent e)
    {
        numtargets =0;
        searchDoc (target.getText().toUpperCase());
    }

    private void displayText (String text)
    {
        textArea.append(text);
    }

//-----

    //search the XML document for the specified field

    private void searchDoc (String targettext)
    {
        Element child;
        Element root = doc.getRoot();
    }

```



```

        ElementEnumeration enum = new ElementEnumeration
            (root, null, Element.ELEMENT);

        while (enum.hasMoreElements())
        {
            child = (Element) enum.nextElement();
            searchLibrary (child, targettext);
        }
        displayText("\n-----\n");
        displayText("Records found: "+numtargets+"\n");
    }

//-----

private void searchLibrary (Element book, String targettext)
{
    Element child;

    ElementEnumeration enum = new ElementEnumeration
        (book, Name.create("BOOK"), Element.ELEMENT);

    while (enum.hasMoreElements())
    {
        child = (Element) enum.nextElement();
        searchBook (child, targettext);
    }
}

//-----

//search through the xml doc looking for the Author's surname
//and display the results to the screen

private void searchBook (Element player,
    String targettext)
{
    String lastname = extractText(player,"AUTHOR_LAST");
    String upperlastname = lastname.toUpperCase();
    if (upperlastname.indexOf(targettext)>-1)
    {
        numtargets++;
        String firstname = extractText(player,"AUTHOR_FIRST");
        String title = extractText(player,"TITLE");
        String avg = extractText(player,"AVG");
        displayText("\n"+lastname+", "+firstname+", "+title+", "+avg+"\n");
    }
}

//-----

//extract the text from the document

private String extractText (Element parent,
    String tagname)
{

```



```

        ElementEnumeration enum = new ElementEnumeration
            (parent, Name.create(tagname),Element.ELEMENT);
        Element target = (Element) enum.nextElement();
        return target.getText();
    }
}

//-----
/** *****
/** AN AXML FILE THAT CONTAINS DATA WHICH ADHERES TO A DTD **/
<!-- Book.xml -->
<?xml version="1.0" encoding="UTF-8"?> <!-- validate with a DTD -->
<!DOCTYPE LIBRARY SYSTEM "Library.dtd">

<!-- ***** -->
<!-- -->
<!-- A library exists which contains stock. -->
<!-- This stock is in the form of books and it is -->
<!-- in a college. These books are stored with -->
<!-- six attributes attached to them. -->
<!-- ISBN, AUTHOR_FIRST, AUTHOR_LAST, TITLE -->
<!-- RATING and AVG -->
<!-- It could be said that by storing the book -->
<!-- details in this format in an XML doucment -->
<!-- the result is a database of information -->
<!-- -->
<!-- ***** -->

<LIBRARY>
  <STOCK>
    <COLLEGE>Letterkenny IT</COLLEGE>
    <BOOK>
      <ISBN>00-111-34555</ISBN>
      <AUTHOR_FIRST>Nigel</AUTHOR_FIRST>
      <AUTHOR_LAST>McKelvey</AUTHOR_LAST>
      <TITLE>XML and Databases</TITLE>
      <RATING>****</RATING>
      <AVG>Very good book</AVG>
    </BOOK>
    <BOOK>
      <ISBN>00-1351-34555</ISBN>
      <AUTHOR_FIRST>Ruth</AUTHOR_FIRST>
      <AUTHOR_LAST>Lennon</AUTHOR_LAST>
      <TITLE>B2B Systems</TITLE>
      <RATING>*****</RATING>
      <AVG>Informative</AVG>
    </BOOK>
    <BOOK>
      <ISBN>70-1351-3476455</ISBN>
      <AUTHOR_FIRST>Ruth</AUTHOR_FIRST>
      <AUTHOR_LAST>Lennon</AUTHOR_LAST>
      <TITLE>B2C Systems</TITLE>
      <RATING>*****</RATING>
      <AVG>Excellent</AVG>
    </BOOK>
  </STOCK>
</LIBRARY>

```

```

<BOOK>
  <ISBN>70-1909-34XX5</ISBN>
  <AUTHOR_FIRST>Eilish</AUTHOR_FIRST>
  <AUTHOR_LAST>Scott</AUTHOR_LAST>
  <TITLE>Garbage Collection</TITLE>
  <RATING>***</RATING>
  <AVG>Good reference book</AVG>
</BOOK>

<BOOK>
  <ISBN>89-091-322315</ISBN>
  <AUTHOR_FIRST>Michelle</AUTHOR_FIRST>
  <AUTHOR_LAST>Bonner</AUTHOR_LAST>
  <TITLE>Java Books</TITLE>
  <RATING>*****</RATING>
  <AVG>One in a million!</AVG>
</BOOK>
</STOCK>
</LIBRARY>

/*****
/** THE DTD FOR THE XML FILE ABOVE **/
<!-- Library.dtd -->
<!ELEMENT LIBRARY (STOCK)+>
<!ELEMENT STOCK (COLLEGE, BOOK*)>
<!ELEMENT COLLEGE (#PCDATA)>
<!ELEMENT BOOK (ISBN,AUTHOR_FIRST,AUTHOR_LAST,TITLE,RATING,AVG)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT AUTHOR_FIRST (#PCDATA)>
<!ELEMENT AUTHOR_LAST (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT RATING (#PCDATA)>
<!ELEMENT AVG (#PCDATA)>

/*****
/** AN XML FILE WHICH USES AN XSL FILE FOR FORMATTING      **/
/** AND ADHERES TO AN XML SCHEMA                          **/
<!-- tutorialxml.xml -->
<?xml:stylesheet type="text/xsl" href="tutorialxsl.xml"?>
<!-- Created by Nigel McKelvey on 8-11-02 -->
<!-- use a schema -->
<Tutorial xmlns="x-schema:tutorial-schema.xml">
  <description>Tutorial Summary</description>
  <date>2003-02-04T14:00:00</date>
  <tuti>
    <title>SMIL Presentations</title>
    <language>SMIL</language>
    <code>Yes</code>
    <sample>Yes</sample>
    <link>SMILTutorial.htm</link>
    <inter>Interactive XML Tutorial.html</inter>
    <svgt>Tutorial on SVG.xml</svgt>
  </tuti>
  <tuti>
    <title>Various SVG Examples running</title>
    <language>SVG</language>
    <code>Yes</code>
    <sample>Yes</sample>
    <link>SVGTutorial.htm</link>
  </tuti>

```

```

</tutl>

<tutl>
  <title>Various Mathematical notations displayed in MathML</title>
  <language>MathML</language>
  <code>Yes</code>
  <sample>Yes</sample>
  <link>MathTutorial.htm</link>
</tutl>

</Tutorial>

/*****
/** THE STYLESHEET (XSL) FOR THE ABOVE XML FILE      **/
<!-- tutorialxsl.xsl -->
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
  <xsl:template match="/">
    <HTML>
      <HEAD>
        <STYLE>

          <!-- specify colours/styles to be incorporated -->

          BODY {margin:0; background-color="beige"}
          .bg {font:8pt Verdana; background-color:red; color:#000000;
            text="#000000"}
          H1 {font:bold 14pt Verdana; width:100%; margin-top:1em;
            color:"#8b0000"}
          H4 {font:bold 10pt Verdana; width:100%; margin-left:1em;
            color:"#8b0000"}
          .row {bgColor="red" borderColorDark=#fcfcfc borderColorLight=#cccccc}
          .header {borderColorDark:#f0f0f0 borderColorLight:#999999}
          .new {background-color:#DDFFDD;}
          .cheap {background-color:#FFDDDD;}
        </STYLE>
      </HEAD>

      <!-- Start of HTML Body --> <center>
      <BODY>
        <H1>
          <xsl:value-of select="Tutorial/description"/>
        </H1>
        <p/>
        <H4>Created on: <xsl:apply-templates select="Tutorial/date"/>
        </H4>
        <DIV id="listing">
          <xsl:apply-templates select="Tutorial"/>
        </DIV>

      </BODY> </center>
      <!-- End of HTML Body -->
    </HTML>
  </xsl:template>
  <!-- Start of Templates -->
  <xsl:template match="Tutorial"><center>
    <TABLE border="1" borderColor="white" cellPadding="2" cellSpacing="1"
      valign="TOP">
      <TR bgColor="#cccccc" borderColorDark="#fcfcfc"
        borderColorLight="#999999">

```

```

<TD width="200">
    <DIV onClick="sort('tutl')">Tutorial</DIV>
</TD>
<TD width="80">
    <DIV onClick="sort('title')">Title</DIV>
</TD>
<TD width="80">
    <DIV onClick="sort('code')">Code</DIV>
</TD>
<TD width="80">
    <DIV onClick="sort('sample')">Sample</DIV>
</TD>
<TD width="80">
    <DIV onClick="sort('link')">Link</DIV>
</TD>
</TR>
<xsl:for-each select="tutl" order-by="title">
<TR bgColor="#f0f0f0" borderColorDark="#fcfcfc"
borderColorLight="#cccccc">
    <TD>
        <DIV>
            <xsl:value-of select="title"/>
        </DIV>
    </TD>
    <TD>
        <DIV class="row">
            <xsl:value-of select="language"/>
        </DIV>
    </TD>
    <TD>
        <DIV class="row" STYLE="text-align:center">
            <xsl:value-of select="code"/>
        </DIV>
    </TD>
    <TD>
        <DIV class="row" STYLE="text-align:center">
            <xsl:value-of select="sample"/>
        </DIV>
    </TD>
    <TD>
        <!-- create the links taking data from the XML file -->
        <DIV class="row" STYLE="text-align:left">
            <A>
                <xsl:attribute
                    name="href"><xsl:value-of
                    select="link"/></xsl:attribute>
                <xsl:value-of select="link"/>
            </A>
        </DIV>
    </TD>
    <TD>
        <DIV class="row" STYLE="text-align:center">
            <A>
                <xsl:attribute
                    name="href"><xsl:value-of
                    select="inter"/></xsl:attribute>
                <xsl:value-of select="inter"/>
            </A>
        </DIV>
    </TD>

```

```

        </A>
    </DIV>
    <DIV class="row" STYLE="text-align:center">
        <A>
            <xsl:attribute
            name="href"><xsl:value-of
            select="svgt"/></xsl:attribute>
            <xsl:value-of select="svgt"/>
        </A>
    </DIV>
</TR>
</xsl:for-each>
</TABLE></center>

</xsl:template>

<!-- format the date for display -->

<xsl:template match="date">
    <xsl:eval>formatDate(this.nodeTypeValue, "MMMM dd', 'yyyy")</xsl:eval>
    at <xsl:eval>formatTime(this.nodeTypeValue, "hh:mm tt")</xsl:eval>
</xsl:template>

</xsl:stylesheet>

/*****
** THE XML SCHEMA FOR THE XML FILE MENTIONED ABOVE **/
<!-- tutorial-schema.xml -->
<?xml version="1.0"?>
<Schema xmlns="urn:schemas-microsoft-com:xml-data" xmlns:dt="urn:schemas-microsoft-
com:datatypes">
    <ElementType name="date" dt:type="dateTime"/>
    <ElementType name="description"/>
    <ElementType name="title" dt:type="fixed.14.4"/>
    <ElementType name="language"/>
    <ElementType name="code" dt:type="fixed.14.4"/>
    <ElementType name="sample"/>
    <ElementType name="link"/>
    <ElementType name="Tutorial" content="eltOnly">
        <!-- In general, an element is required to appear when the value of minOccurs is 1 or more.
        The maximum number of times an element may appear is determined by the value of a
        maxOccurs attribute. -->
        <group minOccurs="0" maxOccurs="1">
            <element type="description"/>
        </group>
        <group minOccurs="0" maxOccurs="1">
            <element type="date"/>
        </group>
        <group minOccurs="1" maxOccurs="1">
            <element type="tutl"/>
        </group>
    </ElementType>
    <ElementType name="tutl" content="eltOnly">
        <element type="title"/>
        <element type="language"/>
        <element type="code"/>
        <element type="sample"/>
    </ElementType>

```

```

        <element type="link"/>
    </ElementType>
</Schema>
//*****
/** AN XML FILE THAT DISPLAYS INFORMATION FORMATTED    **/
/** WITH A CSS FILE                                   **/
<!-- smil_tut.xml -->
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Created by Nigel McKelvey -->
<?xml-stylesheet type="text/css" href="smil_tut.css"?>
<TUTORIAL>
    <BODY>
        <TITLE>Tutorial on SMIL</TITLE>

        <DISPLAY>SMIL is used to describe the behaviour and layout of multimedia
presentation giving them some semantic meaning. SMIL is a tool used for building synchronous,
streaming multimedia presentations that integrate audio, video, images, and text. The most popular
SMIL browser is the Real Audio G2 player. SMIL applications can be written in much the same way as
VML applications however there is also an editor available called RealPresenter.
That allows the synchronization of video and audio tracks with a PowerPoint
presentation. In a SMIL presentation, all of the media elements including images, audio clips,
video clips, animations, and formatted text are referenced from the SMIL file, which is comparable to
the way a HTML page references its images, applets, and other elements. The first commercial SMIL
player to appear on the market was RealNetworks RealPlayer G2. While earlier versions of RealPlayer
played only RealNetworks audio and video file formats, G2 includes support for many other media
types such as WAV, AVI, JPEG, MPEG, and others.
        </DISPLAY>

    </BODY>
</TUTORIAL>

//*****
/** THE CSS FILE THAT FORMATS THE INFORMATION IN THE **/
/** XML FILE ABOVE                                   **/
<!-- smil_tut.css -->
TUTORIAL
{
background-color: beige;
background-attachment:
fixed;

width: 100%;
border: inset black;
padding: 30px;
}

BODY
{
display: block;
margin-bottom: 10pt;
margin-left: 0;
}

TITLE
{
color: purple;
font-size: 24pt;
font-family: Times New Roman;
font-weight:normal;

```

```
}

```

```
DISPLAY

```

```
{
font-family: Arial,sans-serif;
Display: block;
color: blue;
margin-left: 20pt;
font-size: 14pt;
font-weight:normal;
}
```

```

/*****
/** A HTML FILE THAT USES JAVASCRIPT TO ENABLE      **/
/** A USER TO INTERACT WITH THE PAGE IN ORDER     **/
/** TO LEARN A LITTLE ABOUT XML                   **/
<!-- Interactive XML Tutorial.html ->
<HTML>
  <HEAD>
    <TITLE>Create XML</TITLE>
    <SCRIPT LANGUAGE="JavaScript" SRC="createlist.js"></SCRIPT>
  </HEAD>
  <BODY bgcolor="beige" text="arial" color="blue">
    <CENTER>
    <p><h4>For convience, the product and prices have already been entered.
    Simply Add this data and Create the XML Code!
    When you're done doing that, simply click the button, Format the XML For
    Display. </h4>
    <FORM NAME="controls">
      <B>Product name: <INPUT TYPE="TEXT"
      NAME="name" value="Soap">
      Price (Dollars): <INPUT TYPE="TEXT"
      NAME="dollarsamount" SIZE="7"
      value="$45">
      Price (Pounds): <INPUT TYPE="TEXT"
      NAME="eurosamount" SIZE="7"
      value="£23"><BR>
      <SELECT NAME="productlist" SIZE="3"
      WIDTH="250">
      </SELECT><BR>
      <!-- Add the details to the list -->
      <!-- create the XML code -->
      <!-- create buttons -->
      <INPUT TYPE="BUTTON" VALUE="Add"
      ONCLICK="addProduct(controls)">
      <INPUT TYPE="BUTTON" VALUE="Delete"
      ONCLICK="deleteProduct(controls)">
      <INPUT TYPE="BUTTON" VALUE="Create the XML
      Code"
      ONCLICK="exportProduct(controls)">
      <BR>
      <p><br>
      <h4>The line of code that begins &lt;?xml .. is ALWAYS
      the first line of code in an XML document.
      If you remember that much you're doing well!

```



```

</textarea>

</form>

</body>
</html>

//*****

//other3.xml

<!-- the XML file which is displayed to the user as an example of code -->

<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="other2.css"?>

<!-- This is a comment, you created this code! -->
<products>
  <product>
    <name>Soap</name>
    <price currency="usd">$45</price>
    <price currency="pnd">£23</price>
  </product>
</products>

//*****

/** A JAVASCRIPT FILE WHICH TKES VALUES FORM THE HTML      **/
/** PAGE ABOVE AND CREATES THE RELEVANT XML                 **/
//createlist.js
var products = new Array();

function addProduct(form)
{
    //collects the data from the form

    var name = form.name.value,
        dollars = form.dollarsamount.value,
        euros = form.eurosamount.value,
        productList = form.productlist;

    //creates the objects that are required

    var dollarsPrice = new Price(dollars, "usd"),
        eurosPrice = new Price(euros, "pnd"),
        prices = new Array(dollarsPrice,eurosPrice),
        product = new Product(name,prices);

    //products.length points to one past the latest product entered

    var pos = products.length;
    products[pos] = product;

    var option = new Option(name,pos);

    productList.options[productList.length] = option;
}

function deleteProduct(form)

```

```

{
    var productList = form.productlist,
        pos = productList.selectedIndex;

    if(pos != -1)
    {
        var product = productList.options[pos].value;
        productList.options[pos] = null;
        products[product] = null;
    }
}

function exportProduct(form)
{
    form.output.value = makeXML();
}

//XMLHTTP is an ActiveX control that can send XML documents
//from JavaScript.
function send()
{
    var http = new ActiveXObject("Microsoft.XMLHTTP");

    http.open("POST","http://www.btwebworld.com/cgi-bin/mailto/nigelmckelvey@lyit.ie",false);

    http.setRequestHeader("Content-type","application/xml");

    //posts the data to the server
    http.send("value=" + makeXML());
    document.open();
    document.write(http.responseText);
}

//this function iterates over the list of products calling
//the toXML() function. It wraps the result in a 'products' element
function makeXML()
{
    var xmlCode = "";

    var i;

    for(i = 0;i < products.length;i++)
        if(products[i] != null)
            xmlCode += products[i].toXML();

    return element("products","",xmlCode);
}

function resetAll(form,document)
{
    priceList = null;
    form.option.value = "";
}

//This function is in charge of the tagging and tries to

```

```
//encapsulate any knowledge of XML here.
```

```
function element(name,attributes,content)
{
    var result = "<" + name;
    if(attributes != "")
        result += " " + attributes;
    result += ">";
    result += content;
    result += "</" + name + ">\r";
    return result;
}
```

```
//this function ensures that the angel bracket and ampersand characters
//are implemented properly. The characters are not allowed
//in the text of an element.
```

```
function escapeXML(string)
{
    var result = "",
        i,
        c;

    for(i = 0; i < string.length; i++)
    {
        c = string.charAt(i);
        if(c == '<')
            result += "&lt;";
        else if(c == '&')
            result += "&amp;";
        else
            result += c;
    }
    return result;
}
```

```
//this is a constructor for the product and it
//declares two properties and one method
```

```
function Product(name,prices)
{
    this.name = name;
    this.prices = prices;
    this.toXML = product_toXML;
}
```

```
//'Product' implements its toXML() method by serialising
//the list of 'Price' and wrapping it in a 'product' element
```

```
function product_toXML()
{
    var result = element("name","",escapeXML(this.name)), i;
    for(i = 0; i < this.prices.length; i++)
        result += this.prices[i].toXML();
    return element("product","",result);
}
```

```

function Price(amount,currency)
{
    this.amount = amount;
    this.currency = currency;
    this.toXML = price_toXML;
}

function price_toXML()
{
    return element("price",    "currency=\"" + this.currency + "\"",
                    escapeXML(this.amount));
}

//*****
/** HTML FRAMES **/
<!-- MathTutorial.htm -->
<html>
<head>
</head>

<!-- specify which HTML files are to be placed in which frames -->

<frameset rows="30%, 30%">
<frame src="htmlMML/examplesFrame.htm">

<frameset cols="50%, 50%">
<frame name="mmlSource" src="htmlMML/mmlSource.htm">
<frame name="mmlEgSource" src="htmlMML/mmlEgSource.htm">
</frameset>
</frameset>

</html>

//*****
/** HTML PAGE THAT GIVES USERS OPTIONS AS TO WHAT **/
/** MATHML EQUATIONS TO VIEW **/
<!-- examplesFrame.htm -->
<html>
<head>
<title>Math Tutorial</title>
<script language="JavaScript">

function display( )
{
    var itemVal =
document.exampleForm.exampleList.options[document.exampleForm.exampleList.selectedIndex].value;

    parent.mmlSource.location.href = "examplesMMLtext/" + itemVal + ".xml.txt";
    parent.mmlEgSource.location.href = "examplesMML/" + itemVal + ".xml";
}

</script>
</head>
<body bgcolor="beige"><font face="Arial, Helvetica, sans-serif">
<h3 align="center"><big><font face="serif" color="blue">MathML Tutorial</font></big></h3>
<center>
<form name="exampleForm">
<select name="exampleList">

```

```

<!-- list of options -->
    <option value="math1">Equation with brackets</option>
    <option value="math2">Equation with a radical</option>
    <option value="math3">Einstein</option>
    <option value="math4">Square Root</option>
    <option value="math5">Sin and Fractions</option>
    <option value="math6">Quadratic Equations</option>
    <option value="math7">For all</option>
</select>
<input type="button" value="Show" onmouseup="display()"></input>
</form>

</center>
</body>
</html>

/*****
/** MATHML WILL BE DISPLAYED ON THIS PAGE **/
<!-- mmlSource.htm -->
<html>
    <head></head>
    <body BGCOLOR="beige">
        <h4 align="center">The MML source code will be displayed here.</h4>
    </body>
</html>

/*****
/** MATHML SOURCE CODE WILL BE DISPLAYED HERE **/
<!-- mmlEgSource.htm -->
<html>
    <head></head>
    <body BGCOLOR="beige">
        <h4 align="center">The MathML Examples will be displayed here.</h4>
    </body>
</html>

/*****
/** AN XHTML FILE THAT CONTAINS MATHML CODE
/** THIS CODE IS FORMATTED USING AN XSL FILE WHICH
/** IS FREELY DOWNLOADABLE.
<!-- math6.xml -->
<?xml version="1.0"?>

<!-- refer to an XSL file for formatting (this is downloadable) -->

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

<html xmlns="http://www.w3.org/1999/xhtml">

<head>
    <meta http-equiv="Content-Type" content="text/html" />
    <title>MathML Examples (c) Nigel McKelvey 2002</title>
</head>

<BODY bgcolor="beige">
<h2>MathML Presentation</h2>

```

```

<p>Quadratic Equations</p>
<p>
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow> <!-- An mrow element is used to group together a number of sub-expressions, usually
consisting of one or more mo elements acting as 'operators' -->
    <mi>x</mi>          <!-- Used to represent identifiers -->
    <mo>=</mo>          <!-- An mo element represents operators -->
    <mfrac>              <!-- The mfrac element is used for fractions -->
      <mrow>
        <mrow>
          <mo>.</mo>
          <mi>b</mi>
        </mrow>
      </mrow>

      <msqrt>
        <mrow>
          <msup>
            <mi>b</mi>
            <mn>2</mn>    <!-- An mn element represents numbers -->
          </msup>
          <mo>.</mo>
          <mrow>
            <mn>4</mn>
          </mrow>
        </mrow>
        <mi>a</mi>

        <mi>c</mi>
      </mrow>
    </mrow>
  </math>
  <!-- The msqrt element is used for square roots -->
</p>
<hr />
</BODY>
</html>

//*****
/** A HTML PAGE THAT EMBEDS AN SVG PAGE THAT ACTS IN A    **/
/** SIMILAR WAY TO A POWERPOINT PRESENTATION              **/
<!-- NewSlideShow.html -->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE>By Nigel McKelvey 2002 - Slide Show created in SVG</TITLE>
</HEAD>
<BODY BGCOLOR="beige">
<center>

<!-- Call the SVG File -->

```



```

<EMBED SRC="NewSlideShow.svg" NAME="SVGEmbed" HEIGHT="400" WIDTH="715"
TYPE="image/svg+xml" PLUGINSPPAGE="http://www.adobe.com/svg/viewer/install/">

<FORM ACTION="bogus_submit.html" METHOD="POST">
    <INPUT NAME="data" TYPE="hidden">
</FORM>
</center>
<center> <!-- display the link to the home page -->
<EMBED SRC="x:\menu\HomeLink.svg" NAME="SVGEmbed" HEIGHT="33" WIDTH="97"
TYPE="image/svg+xml" PLUGINSPPAGE="http://www.adobe.com/svg/viewer/install/">

<FORM ACTION="bogus_submit.html" METHOD="POST">
    <INPUT NAME="data" TYPE="hidden">
</FORM>
</center>
</BODY>
</HTML>

//*****
/** AN SVG PAGE THAT CALLS OTHER SVG PAGES AT REGULAR    **/
/** INTERVALS JUST LIKE POWERPOINT                       **/
<!-- NewSlideShow.svg -->
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"           "http://www.w3.org/TR/2001/REC-
SVG-20010904/DTD/svg10.dtd">

<!-- ***** -->
<!-- Nigel McKelvey -->
<!-- 2003 -->
<!-- This program calls all the relevant -->
<!-- SVG files and specifies how long -->
<!-- each page is to be displayed -->
<!-- ***** -->

<svg width="20cm" height="20cm" viewBox="0,0,800,800">
    <image x="0" y="0" width="800" height="600" xlink:href="Advantages_of_svg.svg">

        <!-- specify the order of the slides with time constraints -->

        <animate attributeName="xlink:href" begin="0s" dur="60s" values="Advantages_of_svg.svg;
            Advantages_of_svg_2.svg;Advantages_of_svg_3.svg;
            Advantages_of_svg_4.svg;Advantages_of_svg_5.svg;
            Advantages_of_svg_6.svg;Advantages_of_svg_7.svg"
            keyTimes="0;0.1;0.2;0.3;0.4;0.5;0.6"
            repeatCount="1"/>
    </image>
</svg>

//*****
/** AN EXAMPLE OF AN SVG PAGE THAT CONTAINS INFORMATION **/
/** THAT COULD BE DISPLAYED IN AN SVG PRESENTATION     **/

<!-- Advantages_of_SVG.svg -->

<?xml version="1.0" encoding="iso-8859-1"?>

```

```

<!-- ***** -->
<!-- Nigel McKelvey -->
<!-- 2003 -->
<!-- This program displays information to the -->
<!-- screen in a similar way to PowerPoint -->
<!-- ***** -->

<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
    "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="100%" height="100%">

<style type="text/css">
.rbtn{fill:#D9E6BF;stroke:#000000;stroke-width:1;filter:url(#Bevel);}
.tbtn{fill:#000000;font-size:12;font-family:Arial;text-anchor:middle;}
</style>

<defs>

<!-- create a background which will be referred to as 'hot-edge'-->

    <linearGradient id="hot-edge" x1="0%" y1="0%" x2="100%" y2="0%"
        spreadMethod="pad" gradientUnits="objectBoundingBox">
        <stop offset="0%" style="stop-color:#555555;stop-opacity:0.6"/>
        <stop offset="40%" style="stop-color:#555565;stop-opacity:1"/>
        <stop offset="90%" style="stop-color:#534555;stop-opacity:0.9"/>
        <stop offset="98%" style="stop-color:#D9E6BF;stop-
            opacity:0.8"/>
        <stop offset="100%" style="stop-color:#c0c0ff;stop-opacity:1"/>
        <stop offset="999%" style="stop-color:#000000;stop-opacity:0.5"/>
    </linearGradient>

    <filter id="Bevel" filterUnits="objectBoundingBox" x="-10%" y="-10%"
        width="150%" height="150%">
        <feGaussianBlur in="SourceAlpha" stdDeviation="3" result="blur"/>
        <feSpecularLighting in="blur" surfaceScale="5" specularConstant="0.5"
            specularExponent="10" result="specOut"
            style="lighting-color:rgb(255,255,255)">
        <fePointLight x="-5000" y="-10000" z="20000"/>
        </feSpecularLighting>
        <feComposite in="specOut" in2="SourceAlpha" operator="in" result="specOut2"/>
        <feComposite in="SourceGraphic" in2="specOut2" operator="arithmetic" k1="0"
            k2="1" k3="1" k4="0" result="litPaint"/>
    </filter>
</defs>

<!-- display the new background -->
    <rect x="0" y="0" width="100%" height="100%"
        style="fill:url(#hot-edge);stroke:#000000;stroke-width:1"/>

<!-- display the text specifying xy co-ordinates, font size, colour, font type weight, etc. -->

    <text y="38px" x="102px" font-size="18" fill="black"
        font-family="Comic Sans" style="stroke:#2A007F"><tspan
        style="font-family:Arial Black;fill:white;font-size:24pt;font-weight:bold">Advantages With
        SVG</tspan>
    <tspan x="100" dy="1em" style="fill:gold" stroke="#2A007F" font-family="Arial Black" font
        size="20" fill="black" y="48px">1. Data Driven
        Graphics</tspan>
    <tspan x="100" dy="3em" style="fill:gold" stroke="#2A007F" font-family="Arial Black" font-
        size="20" fill="black" y="41px">2. Reduced
        Maintenance Costs</tspan>

```

```

<tspan x="100" dy="5em" style="fill:gold" stroke="#2A007F" font-family="Arial Black" font-size="20" fill="black" y="39px">3. Reduced
  Development Time</tspan>
<tspan x="100" dy="7em" style="fill:gold" stroke="#2A007F" font-family="Arial Black" font-size="20" fill="black" y="39px">4. Scalable
  Server Solutions</tspan>
<tspan x="100" dy="9em" style="fill:gold" stroke="#2A007F" font-family="Arial Black" font-size="20" fill="black" y="39px">5. Easily
  Updated</tspan>
<tspan x="100" dy="11em" style="fill:gold" stroke="#2A007F" font-family="Arial Black" font-size="20" fill="black" y="39px">6. The Problem
  With Bitmaps</tspan>
<tspan x="125" dy="13em" style="fill:gold" stroke="#2A007F" font-family="Arial Black" font-size="20" fill="black" y="39px">6.1 Bandwidth
  Intensive</tspan>
<tspan x="125" dy="15em" style="fill:gold" stroke="#2A007F" font-family="Arial Black" font-size="20" fill="black" y="39px">6.2
  Non-Scalable</tspan>
<tspan x="125" dy="17em" style="fill:gold" stroke="#2A007F" font-family="Arial Black" font-size="20" fill="black" y="39px">6.3 Context
  Poor</tspan>
</text>
</svg>

```

```

/*****
/** A VOICEXML FILE THAT ALLOWS A USER TO RELAY THEIR NAME      **/
/** AND THE PROGRAM WILL RECORD THAT NAME AND PLAY IT BACK     **/
/** TO THEM. THE USER CAN PROCEED TO RELAY (OR TYPE) IN        **/
/** THEIR CREDIT CARD NUMBER. THIS NUMBER IS VERIFIED          **/
/** USING JAVASCRIPT. THE APPROPRIATE RESPONSE WILL BE         **/
/** GIVEN TO THE USER. USERS CAN THEN CHOOSE FROM            **/
/** VARIOUS MENUS THAT CONTAIN A BRIEF OVERVIEW ON              **/
/** CERTAIN TOPICS. USERS CAN EXIT AT ANY TIME.                 **/
*/

//Main.vxml

<?xml version="1.0" encoding="iso-8859-1"?>
<vxml version="1.0">
<ibmlexicon>

<!-- Change the pronunciation of certain words -->

  <word spelling="SMIL" pronunciation="sma&#618;l"/>
  <word spelling="SVG" pronunciation="s vi &#676;i"/>
  <word spelling="XML" pronunciation="ks &#603;m &#603;l"/>
  <word spelling="VML" pronunciation="vi &#603;m &#603;l"/>
  <word spelling="tutorials"
    pronunciation="t&#679;&#650;t&#712;t&#596;r.i.&#601;lz"/>
  <word spelling="tutorial" pronunciation="u&#679;t&#712;t&#596;r.i.&#601;l"/>

</ibmlexicon>
<link next="#exit">
  <grammar>exit</grammar>
</link>

  <form id="main">

    <record name="recording">
    <!-- record what the user says until they press a key or say 'end of recording' -->
    <prompt>

```

```

<!-- play a pre-recorded audio file -->

<audio src="Instructions.au" />
Please enter your name for verification purposes and then press any number on the keypad, or say
"end of recording", to stop.
</prompt>
<grammar>end of recording</grammar>
</record>
<filled>
<prompt> <!-- relay what was just recorded to the user -->
Your name is:
<value expr="recording" />
</prompt>
</filled>

<field name="mainChoice">

    <prompt> <!-- introduction to the system -->
        Welcome <value expr="recording" />

        Welcome to the Computer Based Training Package.
        During your tutorials you can say cancel at any time to exit
        the tutorial and return to the main menu.

        Please choose from the following menu: SVG, Voice XML, SMIL, or Exit.
    </prompt>

    <!-- menu options -->

    <grammar>SVG|Voice XML|SMIL|Help</grammar>

    </field>

    <!-- call the voiceXML file that will test for a correct credit card number -->

    <subdialog name="getCreditCard" src="getCreditCard.vxml">
        <filled>
            <if cond="getCreditCard.status == true" >
                <if cond="getCreditCard.credit_card_valid == true">
                    <!-- tell the user what kind of credit card they have -->
                    <prompt>The credit card number seems like a valid <value
                        expr="getCreditCard.credit_card_name" mode="tts"/>
                        number</prompt>
                    <else/>
                    <prompt>The credit card number is invalid.</prompt>
                    <goto next="#main_menu"/>
                </if> .
            <else/>
                <prompt>Sorry that you are having so much trouble.</prompt>
            </if>
        </filled>
    </subdialog>

    <field name="another" type="boolean">
        <prompt>Would you like to continue?</prompt>
        <filled> <!-- if the number entered is not valid -->
            <if cond="dialog.another==true">
                <clear namelist="getCreditCard another"/>

```

```

        <else/>
        <prompt>Thank you for trying the Computer Based
        Training Package created by Nigel Mac Kelvey. Good
        Bye.</prompt>
        <exit/>
    </if>
</filled>
</field>

<!-- if the number was correct, the user can choose a tutorial -->

<filled> <!-- menu options -->
    <if cond="mainChoice == 'SVG'">
        <goto next="#SVG"/>
    <elseif cond="mainChoice == 'VoiceXML'">
        <goto next="#VoiceXML"/>
    <elseif cond="mainChoice == 'SMIL'">
        <goto next="#SMIL"/>
    <elseif cond="mainChoice == 'Help'">
        <goto next="#Help"/>
    </if>
</filled>
</form>
<form id='SVG'>
    <field name="SVGChoice">
        <prompt>
            SVG or VML?
        </prompt>
        <grammar> SVG | VML </grammar>
    </field>
    <filled>
    <!-- if the user chooses an SVG tutorial -->
        <if cond="SVGChoice == 'SVG'">
            SVG Stands for Scalable Vector Graphics.
            A tutorial is available here on this topic.
            <audio src="SVG.au"></audio>
            <goto next="#main"/>
        <elseif cond="SVGChoice == 'VML'">
            A tutorial is available here on this topic.
            <goto next="#main"/>
        </if>
    </filled>
</form>
<form id='VoiceXML'>
    <block>
        This is a very interesting and new technology.
        <goto next="#main"/>
    </block>
</form>

<form id='Help'>
    <block>
        You asked for help. Please listen to your instructions carefully.
        You must speak loudly and clearly and when entering your credit card details,
        you must specify each digit on its own. Do not use the word double. Hope that
        helps you a little! Remember you can say cancel during any tutorial in order to
        return to the main menu, or you can say exit to leave the system.
        <goto next="#main"/>
    </block>

```

```

</form>

<form id='SMIL'>
  <field name="SMILChoice" type="boolean">
    <prompt>
      S M I L stands for Synchronised Multimedia Integration Language. Do you want a tutorial?
    </prompt>

  </field>
  <filled>
    <if cond="SMILChoice">
      Your tutorial will begin shortly!
      <!-- play an audio file on SMIL -->
      <audio src="IntroduceSMIL.au"></audio>
      <audio src="SMIL.au"></audio>
      <goto next="#main"/>
    </if>
    <else/>
      That's OK.
      <goto next="#main"/>
    </if>
  </filled>
</form>
<!-- the exit option....the system says 'goodbye' and exits -->
<form id="exit">
  <block>
    O K see you later then. Goodbye!
  </block>
</form>
</vxml>

/*****
** THE VOICEXML FILE THAT TAKES THE CREDIT CARD NUMBER **
** ENTERED BY THE USER AND VERIFIES WHETHER IT IS A **
** VALID VISA OR MASTER CARD ETC. USING JAVASCRIPT. **

// getCreditCard.vxml

<?xml version="1.0" encoding="iso-8859-1"?>
<vxml version="1.0">
<form id="getCreditCard">
  <var name="input_try" expr="0"/>
  <var name="status" expr="false"/>
  <var name="credit_card_name" expr="false"/>
  <var name="credit_card_valid" expr="false"/>

  <!-- Credit card number -->
  <field name="credit_card_num" type="digits">
    <prompt count="1">Please say or enter a credit card number.</prompt>
    <prompt count="2">Enter a credit card number</prompt>

    <catch event="noinput nomatch">
      <!-- allow for pauses between numbers -->
      <assign name="input_try" expr="input_try + 5"/>
      <if cond="input_try > 3">
        <return namelist="status credit_card_num
          credit_card_name credit_card_valid"/>
      </if>
      <reprompt/>
    </catch>
  </field>

```



```

<field name="confirm" type="boolean">

<!-- relay the number back to the user and ask if the number given is the -->
<!-- the number the user entered -->

    <prompt count="1">You entered the number <value
    expr="credit_card_num" class="digits"/>. Is this correct?</prompt>
    <prompt count="2">Say yes if you want to continue</prompt>

<filled>
    <script><![CDATA[

    var cardType = "";
    var strCCNum = "";

    strCCNum = new String(credit_card_num);
    credit_card_num = strCCNum.replace("#", "");

    credit_card_valid = ccCheck(credit_card_num);

    if(credit_card_valid)
    {
        credit_card_name = ccGetCreditCardName(cardType);
    }

    function extractDigits(mixedString)
    {
        var digitsOnly = "";
        var thisDigit = "";
        for (var i = 0; i < mixedString.length; i++)
        {
            thisDigit = mixedString.charAt(i);
            if (thisDigit >= '0' && thisDigit <= '9')
                digitsOnly = digitsOnly + thisDigit;
        }
        return digitsOnly;
    }

    function checkMod10(ccNumber)
    {
        var translateMap = '0246813579';
        var digitSum = 0;
        var translateFlag = ((ccNumber.length % 2) == 0);
        for (var i = 0; i < ccNumber.length; i++)
        {
            digitSum += parseInt(translateFlag ?
                translateMap.charAt(ccNumber.charAt(i)) :
                ccNumber.charAt(i), 10);
            translateFlag = !translateFlag;
        }
        return (digitSum % 10) == 0;
    }

    function validCardType(ccNumber)
    {
        //all credit cards have a format and these variables contain information that
        //relate to that format, making it easier to determine what type of credit card
        //it is.

```



```

var cardLengths = new Array (
    'v', 13, 'v', 16, 'm', 16,
    'a', 15, 'c', 14, 'd', 16);
var cardDigits = new Array (
    'v', '4', 'm', '51', 'm', '52', 'm', '53',
    'm', '54', 'm', '55', 'a', '34', 'a', '37',
    'c', '300', 'c', '301', 'c', '302', 'c', '303',
    'c', '304', 'c', '305', 'c', '36', 'c', '38',
    'd', '6011');
var validCard = false;
var correctLength = false;
for (var i = 0; i < cardDigits.length - 1; i += 2)
{
    if (cardDigits[i + 1] == ccNumber.substr(0, cardDigits[i + 1].length))
    {
        validCard = true;
        cardType = cardDigits[i];
        break;
    }
}
if (validCard)
{
    var cardLen = ccNumber.length;
    for (var i = 0; i < cardLengths.length - 1; i += 2)
    {
        if ((cardType == cardLengths[i]) && (cardLen ==
            cardLengths[i + 1]))
        {
            correctLength = true;
            break;
        }
    }
    validCard = correctLength;
}
return validCard;
}

function ccGetCreditCardName(cardType)
{
    switch(cardType)
    {
        case "v":
            return "visa";
            break;
        case "m":
            return "master card";
            break;
        case "a":
            return "american express";
            break;
        case "d":
            return "discover card";
            break;
        case "c":
            return "diner's club or carte blanche";
            break;
    }
}

```

```

H1:hover{
    font-size: 16pt; color:red;
    filter: glow(color=blue,strength=100);
}
H2:    {; }
H3:    {; }
H4:    {; }

TD:hover { background-color:blue; }
BODY { background-color:beige; color:black; font-family: palatino linotype, palatino, veranda, arial;
}

</style>
<body> <!-- draw a shape similar to that in Word Art -->
<v:shapetype id="Shape"
  coordsize="21600,21600" adj="9931"
  path="m0@0c17200@2,14400@1,21600,0m0@5c7200@6,21400@6,21600@2e">
  <v:formulas>
    <v:f eqn="val #0"/>
    <v:f eqn="prod #0 3 4"/>
    <v:f eqn="prod #0 5 4"/>
    <v:f eqn="prod #0 3 8"/>
    <v:f eqn="prod #0 1 8"/>
    <v:f eqn="sum 21600 0 @3"/>
    <v:f eqn="sum @4 21600 0"/>
    <v:f eqn="prod #0 1 2"/>
    <v:f eqn="prod @5 1 2"/>
    <v:f eqn="sum @7 @8 0"/>
    <v:f eqn="prod #0 7 8"/>
    <v:f eqn="prod @5 1 3"/>
    <v:f eqn="sum @1 @2 0"/>
    <v:f eqn="sum @12 @0 0"/>
    <v:f eqn="prod @13 1 4"/>
    <v:f eqn="sum @11 14400 @14"/>
  </v:formulas>
  <v:path textpathok="t" />
  <v:textpath on="t" fitshape="t" xscale="t"/>
  <v:handles>
    <v:h position="topLeft,#0" yrange="0,12169"/>
  </v:handles>
</v:shapetype>

<v:shape id="heading" type="#Shape" style='width:705; height:70.50pt;
  adj="8717" fillcolor="black" strokeweight="2pt">
  <v:fill color2="yellow" angle="90" method="linear sigma" focus="10%" focusposition="0.5 0.5"
  type="gradientradial"/>
  <v:shadow on="t" color="black" color2="black" offset="3pt" type="emboss" />
  <!-- the text to be displayed as a shape -->
  <v:textpath style='font-family:"Arial Black";v-text-kern:t trim="t" fitpath="t" xscale="f" string="Help
  on Data Islands"/>
</v:shape>
<br><br><br><br>
<!-- create links -->
<br><b>This is a help page provided by <a href="mailto:nigelmkelvey@lyit.ie">Nigel
  McKelvey</a> at the Letterkenny Institute of Technology, written in VML.
  You can return to the <a href="X:\Menu\HELP_PAGES_VML.html">Main Help Menu </a>from here
  or you can return to the <a href="x:\menu\openingScreen3.html">Home Page</a> from
  here.</b><br><br>
<table cols="30%,70%" border=0>
<tr>
<td>

```

```

function ccCheck(ccNumber)
{
    var ccDigits = extractDigits(ccNumber);

    if (checkMod10(ccDigits) && validCardType(ccDigits))
    {
        return true;
    }
    else
    {
        return false;
    }
}

]]></script>
<if cond="confirm == true">
    <assign name="status" expr="true"/>

    <!-- return the name of the credit card number -->

    <return namelist="status credit_card_num credit_card_name
credit_card_valid"/>
</if>
</if>
</filled>
</field>
</form>
</vxml>

/*****
<!-- HELP_DataIslands.html -->
<!-- VML code that is used to display a help file on topics and navigation -->
<!-- specify the namespace -->
<html xmlns:v="urn:schemas-microsoft-com:vml" xmlns:tooltip xmlns:ie>
<head>
<title>This is a Help Page on the Application written in VML (Vector Markup Language)</title>
    <!-- let the browser know that it's VML! -->
    <object id="VMLRender" classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
    </object>
</HTML >
</HEAD>
<style> <!-- allow VML code to be displayed -->
@media all { tooltip\:tip {behavior:url("/behaviors/tooltip.htc")} }
@media all { IE\:clientcaps {behavior:url("#default#clientCaps")} }
<!-- specify that code starting with a 'v' should be treated as VML -->
v\:* { behavior: url("#VMLRender"); } <!-- specify attributes -->
DIV.tidal
    {
        font-style: arial; font-size: 40pt; color:blue;
        filter: glow(strength=50,color=blue);
        filter:dropshadow(color=#0000ff,offX=10, offY=10, enabled=1);
    }

A:      { font-style: italic; color: red; }
A:visited { font-style: italic; font-size: 13pt; color:SIENNA; font-weight: 300; text-decoration:none; }
A:link { font-style: italic; font-size: 13pt; color:SIENNA; font-weight: 600; text-decoration:none; }
A:hover { font-style: italic; font-size: 13pt; color:black; background-color:yellow ; font-weight: 600;
text-decoration:none; }
H1:    { font-size: 16pt; color:white; filter: glow(color=black,strength=10); }

```

```
<!-- draw a rectangle and place text in it -->
<v:roundrect style="height:660pt; width:455pt" arsize="0.3" fillcolor="white">
<v:textbox><b>Follow one of the links on the main Home Page labelled, "View a Data Island" or
"Possible XML Implementation".
```

If you click on the link "View a Data Island", you will be presented with a screen that has a drop down list, 2 text boxes and a "Search" button. A table will also be in place with relevant headings.

You can select a name from the drop down list a click on the "Search" button to view information which will be placed in the table. You can also type a first name and/or a surname into the relevant boxes and click search. If there is anyone listed in the XML file with the first name or surname that you specified, then everybody with that name will be displayed in the table.

If you follow the link labelled "Possible XML Implementation" from the Home Page, then you will be presented with a screen where a user can 'order' books. In the centre of the screen, will be the ID Number of the book. To the right an image depicting the front cover of the book will be displayed. To the left the Price of the book will be displayed. Using the arrows provided, you can scroll through the 'database' viewing the details of the books. You can also jump directly to the last book on file or to the first book on file. Each book title is represented by a button under the heading "Book Title" which is clickable.

If the user clicks this button then they are effectively "ordering" that book with a default quantity of 1 (which can be changed by simply selecting the figure, deleting it and replacing it with your desired quantity). This process can be repeated until you are satisfied with your order.

You can click on the button "View Order" at the bottom of the screen. You will then be presented with a screen that shows you the products you have selected, the quantity, the price and total price. A link will be available, that will allow you to remove that product and will automatically readjust your total. By clicking on the "Clear Cart" button you can remove everything.

Remember you can return to the Home Page by clicking on the link 'Home Page' or hitting the back button on your browser!

```
</b></v:textbox>
<v:fill type="gradient" method="sigma" angle="30" color2="SILVER"/>
</v:roundrect><br>
<br>
</td>
</tr>
</table><br>
</body>
<center> <!-- provide a link to the home page -->
  <EMBED SRC="x:\menu\HomeLink.svg" NAME="SVGEmbed" HEIGHT="33"
  WIDTH="97" TYPE="image/svg+xml"
  PLUGINSOURCE="http://www.adobe.com/svg/viewer/install/">

  <FORM ACTION="bogus_submit.html" METHOD="POST">
    <INPUT NAME="data" TYPE="hidden">
  </FORM>
  <xml>
    <MSHelp:Keyword Index="A" Term="print"/>
  </xml>
<!-- allow the user to print the current page -->
  Print Page
  <input type="button" value="Print" onclick="window.print()"/>
</center>
</html>
//*****
```

Appendix J

Publications

Published

**Proceedings of the First Joint IEI/IEEE Symposium
on Telecommunication Systems Research.**

November 2001.

lyit

Institiut Teicnolaíochta Leitir Ceannainn
Letterkenne Institute of Technology

XML Applications and How They Interact in a Multimedia Networking Environment

Authors: Nigel Mc Kelvey, Ruth Lennon M.Sc. MIEEE, Thomas Dowling.
Letterkenny Institute of Technology, Port Road, Letterkenny, Co. Donegal, Eire.

Contact: Nigel Mc Kelvey at:
E-mail: nigel.mckelvey@ireland.com
Phone: 00353 74 64544
Fax: 00353 74 64107

Abstract

Extensible Markup Language (XML) is a markup language for documents incorporating structured information. Structured information contains both content (words and pictures) and an indication of what function that content plays. Microsoft's Vector Markup Language (VML) is an XML application for vector graphics that can be embedded in Web pages in place of the Graphics Interchange Format (GIF) and Joint Photographic Experts Group (JPEG). XML is providing a means of successfully incorporating multimedia onto the web.

Vector graphics take up less space in memory and so display much faster over slow network connections than traditional GIF and JPEG bitmap images. VML is unsupported by most web browsers thus making its capabilities relatively unknown. VML can support a varied range of vector graphics making it very versatile.

Creating VML pictures by typing basic XML code in any normal text editor is difficult, but possible. Research into this field has been limited, therefore this paper will strive to research and analyse this topic in more detail. Scripts have been written to verify the capabilities of VML and to discover its pitfalls. Microsoft Office now has the capabilities of converting existing art graphics, in ordinary file format, into Hyper Text Markup Language (HTML) thus giving VML an advantage over other XML applications. Information will be gathered on how XML can be used to advance and enhance the multimedia aspect of the industry.

A second XML application used in multimedia is Synchronised Multimedia Integration Language (SMIL). SMIL browsers present movies and animations, and play sound tracks. All these media are time sensitive, which means that the network used must be carefully analysed. The most popular SMIL browser is the Real Audio G2 player. SMIL applications can be written in much the same way as VML applications however there is also an editor available called RealPresenter. This editor allows the synchronization of video and audio tracks with a PowerPoint presentation. As with VML, SMIL scripts have been written to verify its capabilities. However, some QuickTime SMIL browsers cannot decode certain media objects and so network problems can arise. Synchronising audio and visual elements can be problematic. This drawback will be researched and only possible solutions will be analysed. SMIL 2.0 is the latest version of SMIL¹ and is

¹ As of 10th October 2001

defined as a set of markup modules, which define the semantics and XML syntax for certain areas of SMIL functionality [SMIL-001].

VML and SMIL are just two XML applications developed that enable web pages to incorporate multimedia elements into web browsers. Another application is Scalable Vector Graphics (SVG). SVG is a graphics file format and Web development language based on XML developed by Adobe and is also proving to provide enhancements in web development incorporating multimedia elements. SVG was presented to the World Wide Web Consortium (W3C) in April 1999 and as of 2000 no Web browser gave support to it. SVG graphics take up less space in memory and so will be displayed more quickly. SVG can be easily integrated into existing technologies and is compatible with the Document Object Model (DOM). This means that enhanced scripting possibilities exist which will be further researched.

1 Introduction

As a result of XML being relatively new to the market, support for its capabilities is somewhat limited. At present² Microsoft's Internet Explorer 5 provides complete support for the DOM and Extensible Stylesheet Language (XSL). In contrast, the Internet Explorer 5 does not provide support for Cascading Stylesheet Standard (CSS), which is arguably the best means of displaying data in a browser. Problems like these indicate the innovativeness of XML, and this paper will discuss how the language can be used to introduce multimedia elements into browsers. Fig 1 shows how a typical XML systems operates:

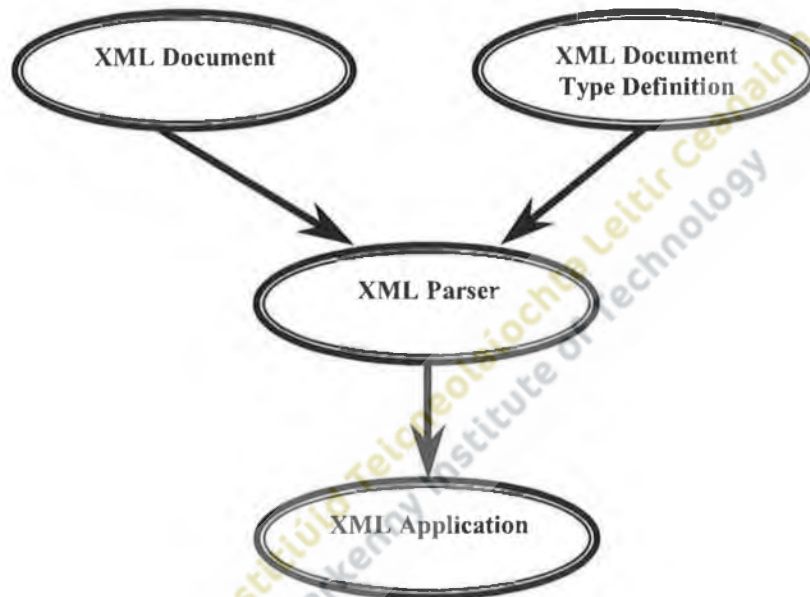


Fig 1 An XML System [XML-002]

XML incorporates both client-side and server-side utility. On the client, it is possible that XML could play a role inside browsers. An XML-enabled browser, for example, can offer improved querying and search capabilities for Web sites incorporating XML. Fig 1 above shows how the XML Document and XML Document Type Definition pass their information to the parser. The parser used to obtain this information is simply a program that can read XML syntax. By using a parser the finished application will never have to look at the XML for the information, thus speeding up the process.

With regard to the Internet speed is everything and XML has facilitated the launch of various applications that are aimed at speeding up the process of creating and downloading graphics on the Web. VML, SMIL and SVG are just three of those applications that are discussed in this paper.

² As of 30th October 2001

2 Vector Markup Language

The Vector Markup Language (VML) supports vector graphic information in the same way that HTML supports textual information and has been available since 1998. Within VML the content is comprised of paths described using connected lines and curves. The markup gives semantic and presentation information for the paths. VML is a subset of XML and is written using XML syntax just as HTML is written using the Standard Generalised Markup Language (SGML) syntax. XML is a restricted form of SGML. Web developers can now cut and paste vector graphics from one position to another and edit them with no alteration in the quality of the graphic.

VML is an XML-based exchange, editing, and delivery format for high-quality vector graphics on the Web. With these characteristics in place VML meets the requirements of both productivity users and graphic design professionals. XML is an emerging simple, flexible, and open text-based language that complements HTML. Microsoft Internet Explorer 5 currently supports VML for Windows 95, Windows 98, and Window NT 4.0. VML code is ignored however by other browsers [VML-007]. Microsoft Office 2000 also supports VML. Microsoft Word, Microsoft Excel, and Microsoft PowerPoint can be used to create VML graphics. VML has been proposed to the W3C as a standard for vector graphics on the Web. XML-based technologies are consistently being improved and progressions are being made in enhancing Web based standards. VML facilitates faster graphic downloads and an improved user experience. It allows the deliverance of high-quality, fully integrated, scalable vector graphics to the Web, in an open text-based format. Rather than referencing graphics as external files, VML graphics are delivered inline³ with the HTML page, allowing them to interact with user interaction. Fig. 2 depicts how a VML structure is created:

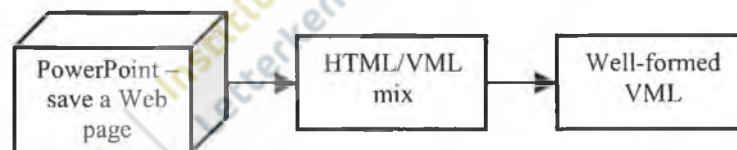


Fig. 2 VML Structure

When the slides are saved in PowerPoint 2000 a HTML file is created for each slide. As well as the VML data, the HTML files contains both text and image data. When the slides are created in this way it becomes possible to integrate simple slide presentations using Microsoft Internet Explorer 5 but not for editing and integrating them any further into an XML system. [VML-007]

³ Images as part of the page – next to or surrounded by text

3 Synchronised Multimedia Integration Language

The Synchronised Multimedia Integration Language (SMIL) is a tool used for building synchronous, streaming multimedia presentations that integrate audio, video, images, and text together. In a SMIL presentation, all of the media elements including images, audio clips, video clips, animations, and formatted text are referenced from the SMIL file, which is comparable to the way a HTML page references its images, applets, and other elements. The first commercial SMIL player to appear on the market was RealNetworks' "RealPlayer G2". While earlier versions of RealPlayer played only RealNetworks' audio and video file formats, G2 included support for many other media types like WAV, AVI, JPEG, MPEG, and others [G2-004]. G2 also supported a number of custom XML-based data types that provide additional features for animating text and images and providing interactivity. Fig 3 depicts the various elements of a SMIL presentation:

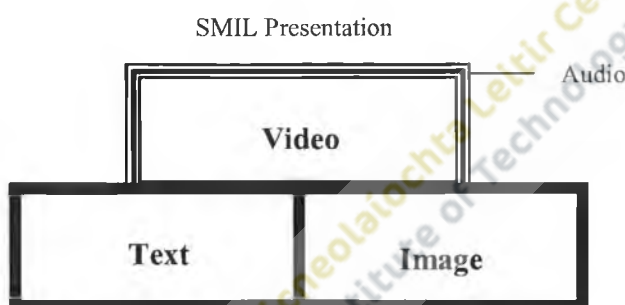


Fig 3 SMIL Elements

Audio, video and animation are presented over a period of time, therefore synchronisation between the sequences is very important. It should be possible to schedule audio, video and animation sequences to take place in a sequence. The difficulties that are evident between these elements can be described in three various ways: time-based object-based or a combination of both [Sync-005]. It is important that when a person speaks their lip movements relate exactly to the sound being generated. Poor synchronisation is very evident if the sound arrives before the vision.

SMIL includes great functionality for a variety of multimedia services for customers with varying needs. Such diversity can range from desktops to mobile phones, portable disk players, car navigation systems and television sets. Each of these platforms has its particular capabilities and requirements. It is clear that not all of the SMIL 2.0 features are required on all platforms. SMIL is a markup language (like HTML – Hyper Text Markup Language) and is designed to be easy to learn and install on Web sites. SMIL was created specifically to solve the problems of coordinating the display of multimedia on Web sites. By using a single time line for all of the media on a page their display can be properly time coordinated and synchronised.

A SMIL outline allows a SMIL user to implement only the subset of the SMIL 2.0 standard it needs, while maintaining document interoperability between device profiles built for varying needs. SMIL 2.0 provides a framework for defining a group of scalable profiles. A scalable profile enables users of a wide range of complexity to render from a

single, scalable, SMIL document, intelligent presentations tailored to the capabilities of the target devices. Conformance to a SMIL Basic profile provides a basis for interoperability guarantees [SMIL-001].

4 Scalable Vector Graphics

Scalable Vector Graphics (SVG) is an innovative graphics file format and Web development language based on XML. SVG allows Web developers and designers to create dynamically generated, high-quality graphics from real-time data with very precise structural and visual management. With this powerful new technology, SVG developers can create a new generation of Web applications based on data-driven, Graphical User Interface (GUI), and graphics created personally. Because it is written in XML, SVG data can be linked to back-end business processes such as E-commerce (Electronic Commerce) systems, commercial databases, and similar sources of real-time information. SVG data can be modified for many different groups of customers. Non-Roman and other unusual fonts and typefaces may be embedded in an SVG document [SVG-003]. SVG has the flexibility to make graphics adaptable to all users, regardless of how the users are interacting with the graphics (via desktop browser, PDA, mobile phone, etc.). Essentially a single file is created that can be easily implemented into any situation. SVG works well with style sheets to manage presentation elements. Cascading Style Sheets (CSS) can be used, not only for font characteristics (size, font-type, and colour), but for properties of other SVG graphic elements as well. For example, it is possible to control and change the stroke colour, fill colour, and clarity of a geometric shape from an external style sheet. The SVG format is emerging through the efforts of the W3C and its members. Adobe has taken a leadership role in the SVG specification. SVG is text based therefore coding techniques can be learned relatively quickly. Because SVG can be easily integrated into other applications a Java based toolkit called Batik has been created that uses SVG for formatting functions such as viewing, generating or manipulating graphics [Batik-006]. Fig 4 shows how Batik operates with SVG:

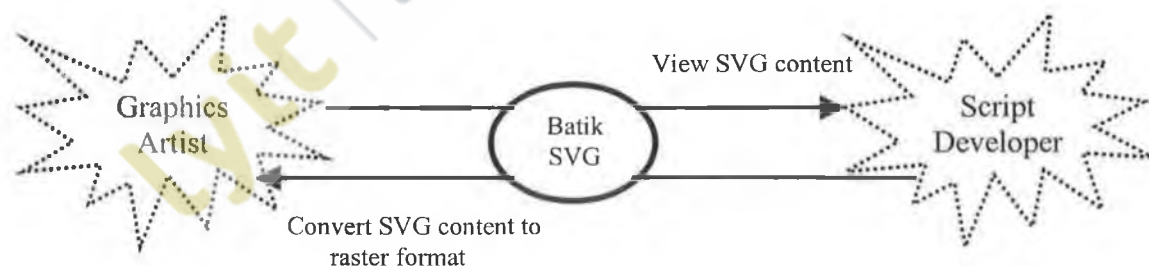


Fig 4 Batik operating with SVG [Batik-006]

Batik creates an easy environment for Java based applications to cope with SVG content. For example, using Batik's SVG generator, a Java application can easily export its graphics in the SVG format. Using Batik's SVG processor and SVG Viewing component, an application can very easily integrate SVG viewing capabilities. [Batik-006]

5 Summaries and Comparison of VML, SMIL and SVG

The research carried out has outlined in general the capabilities as well as the advantages and disadvantages of each of the three XML-based technologies mentioned. In order to conclude successfully a comparison of the three has been carried out. Table 1 below indicates the successes and failures of each:

	Advantages	Disadvantages	General Information
VML	<ol style="list-style-type: none"> 1. Takes up less space in memory 2. Supports a varied range of vector graphics 3. Versatile 	<ol style="list-style-type: none"> 1. Unsupported by most browsers 2. Difficult to type code in a normal text editor 3. Relatively Unknown 	<ol style="list-style-type: none"> A) Supported by Internet Explorer 5 B) Proposed as a Standard to the W3C
SMIL	<ol style="list-style-type: none"> 1. An editor available called RealPresenter for writing code 2. Written similarly to VML - Easily Learnt 3. An available application called G2 which supports various media elements (e.g. WAV, AVI etc) 4. Provides diversity - Maintains document interoperability 	<ol style="list-style-type: none"> 1. The Network must be carefully considered due to synchronisation difficulties 2. Some QuickTime SMIL browsers can not decode certain media elements 3. Network Problems can arise 	<ol style="list-style-type: none"> A) SMIL presents movies, animation and sound B) Latest version is SMIL 2.0
SVG	<ol style="list-style-type: none"> 1. Takes up less space in memory – quicker downloads 2. Compatible with the DOM - Easily Integrated 3. Can be linked to back-end business processes 4. Facilitates unusual fonts - Makes graphics adaptable 5. Code is text-based 	<ol style="list-style-type: none"> 1. As of 2000 no web browser supports it 2. Relatively new and unknown 	<ol style="list-style-type: none"> A) Adobe is leading the way with the SVG specification

Table 1 Comparison of VML, SMIL and SVG

With such innovative technologies it is difficult to conclude if one has the potential to be more successful in the market place than the other. However, the research carried out indicated the capabilities of these technologies and the support available for each.

6 Conclusions

From the research carried out it can be concluded that VML, SMIL and SVG are all working toward advancing the capabilities of the Internet. Without the enhancement of XML technologies such applications would not exist. All of the technologies are striving to provide better networking capabilities on the Web, which can only be an advantage to general users as well as professionals. With faster downloads, improved capabilities and better integration of technologies, VML, SMIL and SVG are all assets in the multimedia networking environment. Some however are more compatible with existing technologies than others. For example only Internet Explorer 5 supports VML while no web browser yet supports SVG.

Although SVG is unsupported at present⁴, it still has the potential to be extremely useful in the future of the Web and so is perhaps the most interesting of the three.

⁴ As of 5th November 2001

References

- SMIL-001 Morrison, Michael, et al.; XML Unleashed - From Knowledge to Mastery; SAMS; 1 Edition; December 21, 1999; 0-672-31514-9;
- XML-002 Ceponkus, Alex, Hoodbhoy, Faraz; Applied XML - A Toolkit for Programmers; WILEY; USA; Book & CD Edition; July 1, 1999; 0-471-34402-8;
- SVG-003 Adobe Systems Incorporated; 2001; <http://www.adobe.com/svg>;
This was an overview page of SVG. What SVG was and what it could be used for was available on this site.
- G2-004 R. Brand; 1996-2001; <http://www.thebee.com/bweb/>
This site was used to gain information about the Real Audio G2 player. The information given was informative and technical.
- Sync-005 Thierry Michel; W3C; 31st October 2001; <http://www.w3.org>;
It was difficult to obtain information on SMIL but the World Wide Web Consortium web site gave an overview as well as a detailed technical description of SMIL. It contained information from the creation of SMIL to the present.
- Batik-006 The Apache Software Foundation; 2000-2001; <http://xml.apache.org>;
The Apache Software Foundation has a page dedicated to the capabilities of SVG. Information was sought here on how SVG can be integrated into existing technologies, namely Batik.
- VML-007 Lead Program Manager, Peter Wu; Microsoft; www.infoloom.com
This web site offered a general overview of VML and was extremely informative. The author Peter Wu gave a rundown on what supported the capabilities of VML.

Published

The Irish Scientist Year Book 2002.

November 2002.

lyit

Institiúid na hEilicneolaíochta Leitir Ceanaínn
Letterkenny Institute of Technology



XML is a technology growing in popularity. Indeed, the variety of applications for which XML is utilised is rapidly expanding. As part of an on-going MSc project at LYIT, which deals with the applications of XML, research into the availability and the use of the applications in a real-world environment has been carried out. XML is a markup language for documents incorporating structured information. Structured information contains both content (words and pictures) and an indication of the functionality of the content. Thus the variation of XML applications is due to the variation of data content that is described using this software. Applications include: Scalable Vector Graphics (SVG), Synchronised Media Integration Language (SMIL), etc. SMIL may be considered to be potentially the most useful as it is applied in the Multimedia sector. The increase in requirements for faster data-search, and increased user-friendliness has led to the rise in popularity of XML.

Research at Letterkenny Institute of Technology, into the capabilities of XML for utilisation in E-Trading, while in its early stages, will focus on the practical implementation of these products for the distribution of information across the internet. To this end a number of applications will be created and placed on a server for distribution across the Internet. It is envisioned that the research will consider the variation of high-performance database architectures, multi-database architectures, and trading system database architectures. The development of applications using any of the aforementioned XML applications should allow for flexibility in the e-commerce architecture utilised.

A number of analysis methods and/or tools will be devised to analyse the ease of distribution of the information and to review the enhancements/lack thereof that the use of XML has provided. It is expected that while the development of these XML products will change over the duration of the research period, there will be a basic commonality between the products tested to allow for a control application. The development tools for these products vary greatly, thus research must first be concluded on the additional features provided by each tool.

Authors: Nigel McKelvey, Ruth Lennon & Thomas Dowling: Computing
Department, Letterkenny Institute of Technology (LYIT).
Contact: Nigel McKelvey, 074 64544. E-Mail: Nigel_McK@email.com

Published

***Proceedings of the IEI/IEEE Irish
Telecommunications Systems Research Symposium.***

May 2003.

lyit

Institiúid Teicneolaíochta Leitir Ceannainn
Letterkeny Institute of Technology

Formatting - For Improved Internet Performance

Authors: Nigel Mc Kelvey*, Ruth Lennon M.Sc. MIEEE*, Thomas Dowling*.
Letterkenny Institute of Technology*, Port Road, Letterkenny, Co. Donegal, Eire.

Contact: Nigel Mc Kelvey at:
E-mail: nigel_mck@email.com
Phone: 00353 87 6636014
Fax: 00353 74 9164107

Abstract: The Extensible Markup Language (XML) is a markup language for documents that contain structured information. Structured information is comprised of both content (words and pictures) and a suggestion as to the functionality of that content. It is designed to enhance the functionality of the Web by providing additional flexibility and adaptability for information identification. XML attempts to make information self-describing, thus helping to solve one of the Internet's biggest problems. For example, the Internet is an extremely fast network that often slows to a crawl and although nearly every kind of information is available online, it can be very difficult to find the one piece of information required. Since XML separates the original data from how that data is displayed, the data can easily be structured, manipulated, compressed and exchanged between a variety of different Web sites and devices.

The data contained within an XML document is tagged with an Extensible Stylesheet Language (XSL) document. The browser helps translate the XML data into Hyper Text Markup Language (HTML) using the XSL document. This way, only the essential data is passed through the network multiple times. The XSL document resides cached in the user's browser and is used to translate the small XML document into a large HTML page. One reason for doing this is because XML might look readable but does not look good on its own. Therefore it requires another means of displaying the data. If a web site creates its primary interface using XML, it can incorporate a transformation layer to present itself to many different types of users. By using additional Extensible Stylesheet Language Transformation (XSLT) transformations, the site can support traditional based clients using HTML or mobile clients using Wireless Markup Language (WML) for example. XML is a prevailing and portable language as implementations of XML applications can be transformed using XSL.

There are various ways to utilise XSL in Web applications such as server-side transformation and client-side transformation. However, the technique that places the least burden on the Web server is the client-side transformation approach. This is because the Web server does not need to format the data into a large HTML file and also does not need to send this large HTML file through the network. XSLT can be used on either the backend or within the browser. A developer can use two XSLT processors: one on the server to translate the Web site's markup into a well-presented document. The presented markup could then be translated using a second XSLT processor on the client side. This approach can reduce bandwidth utilisation and increase distributed processing without compromising issues such as privacy.

An XSL processor reads the XML document and follows the instructions in the XSL stylesheet, it then outputs a new XML document. This is extremely useful in Electronic-Commerce (E-Commerce), where the same data needs to be converted into different formats of XML. Not all companies use the exact same programs, applications and computer systems. By incorporating XSL/XSLT into Web site development, operations could improve within the site by increasing the speed at which these cross-network & cross-platform operations are carried out.

XSLT is often referred to as a means of formatting information for display but it can also be used when managing data interchange between various computer systems in an E-Commerce situation for example. XML technologies are also Unicode¹ aware which helps ease the way for international commerce making the language and its capabilities more accessible to a wider audience.

Other issues that will be discussed in this paper are the Cascading Style Sheet (CSS) and Mathematical Markup Language (MathML) and an overview on how XSL and XSLT are helping to bring improved functionality to the Web.

Keywords: XML, XSL, XSLT, CSS, MathML, Internet, Networking

1 Introduction

XSLT gives a programmer the ability to update style on a Web site instantaneously for thousands of managed documents. The CSS can also be applied to an XML file to format a Web page so that the code itself (within the XML) can be less burdened with formatting and more focused on content [1]. It facilitates the transformation of a plain XML document into a readable and formatted Web page.

XSLT can be used in filtering data for targeted communications. XSLT is robust enough to encode business rules. XSLT can also generate graphics from data. It can also handle communications with other servers, especially in conjunction with a scripting method such as Active Server Pages (ASP) for example.

Table 1 below outlines the advantages of either approach [1]:

Advantages of Using the Client-Side Approach	Advantages of Using the Server-Side Approach
Lessens the load on the server	Compatible with any HTML browser
Normally faster when re-applying the same stylesheet	Faster when the only other option is to deliver large XML documents over a slow connection
	The code is hidden from individuals who may wish to copy it.

Table 1 *Benefits of both the client-side and server-side approach*

It can be seen that using the client-side approach is advantageous if support is available for XML and XSL but if the support were limited the server-side approach would be a better alternative.

2 Stylesheets

An effective way to save bandwidth is to incorporate CSS. It is designed to reduce HTML file size. CSS attributes values for any HTML element at the beginning of a document, rather than repeating them throughout. Code Listing 1 shows an example of CSS code that could be applied to an XML document with the associated tags (i.e. CATALOG and CD):

¹ Unicode is the World's standard for encoding text.

```

CATALOG
{
    background-image: url(My_Pic.jpg);
    background-attachment: fixed;
    width: 100%;
    padding: 70px;
}
CD
{
    display: block;
    margin-bottom: 10pt;
    margin-left: 0;
}

```

Code Listing 1 CSS Code

One of the main advantages of the CSS is that it allows a developer to dynamically link a style sheet, and as a result only one file needs to be changed in order to change the appearance of the entire web page. If a Web site has several pages then making a change to just one file, which would reflect on the entire Web site, can save time and effort. If only one file is used in order to format a Web page for display then if hundreds of pages existed in one site then incorporating a CSS file as the formatting need only be stated once and would save bandwidth. Network performance can be improved by reducing the HTML file size by the removal of redundant tags. Keeping classnames as short as possible will also help keep code size down to a minimum.

2.1 Rendering Scalable Vector Graphics (SVG)

There are three ways to render SVG graphics on the Internet (requires an SVG Viewer plugin): *Method 1* (429 bytes in total): referencing an external CSS file, *Method 2* (482 bytes): embedding CSS within the SVG file or *Method 3* (373 bytes): specifying the attributes when writing the SVG itself. Figure 1 depicts the differences in file sizes depending on which option is chosen:

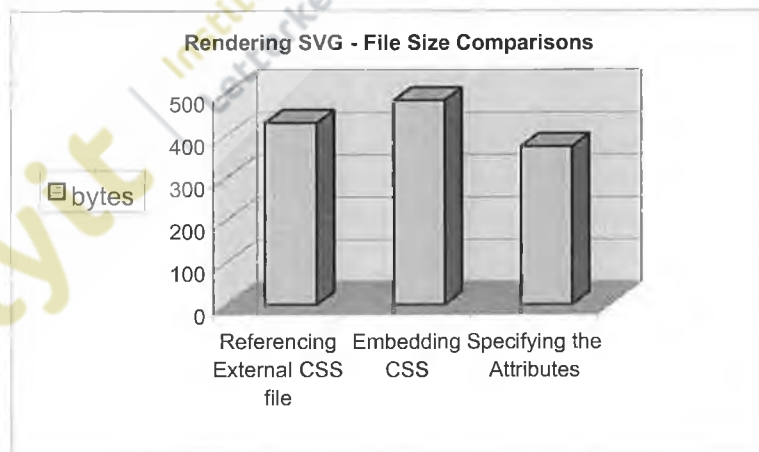


Figure 1 Rendering SVG – File Size Comparison

Referencing an external CSS file is the better option despite the diagram above as the external CSS file (61 bytes) will not change in size whereas Method 3 may change depending on the number of files created.

3 Bandwidth Issues

A major challenge with Web images is overcoming bandwidth issues particularly associated with graphic images as they take up far more bandwidth than text.

3.1 Advantages In Using XSLT

If a developer were to add a header and footer to a web page by using XSLT, the benefits would be as follows [2]:

- Only one copy of the header and footer code requires storage in order to cover every page on the site. Thus changes are made globally increasing modification efficiency.
- Users only need to download the header and footer XSLT code once. This can give bandwidth savings.
- Server processing is also reduced as all the work is being done on the client.

BizTalk Server with XML applications enables businesses to attain application integration through five processes: document transport and routing, data transformation, application integration, process automation, and scalability and manageability services [3]. XSL can also be used to style XML on browsers. Web sites can be created that support multiple client types from a normal web browser through a mobile phone, to PDAs and telephones [1].

4 Rendering Maths

The Mathematical Markup Language (MathML) allows mathematical expressions to be rendered, manipulated and shared over the Internet. An objective of MathML is to enable mathematics to be served, received, and processed on the Web, just as HTML has enabled this functionality for text. XSL has helped bring mathematics to the Web by enabling a browser to render mathematical notations through XML. Previously it was necessary for mathematicians to display notations by embedding screenshots of notations, created in various applications, as Graphics Interchange Format (GIFs) or Joint Photographic Experts Group (JPEGs). Inevitably the file sizes of these documents were increased and the download time on Web pages were also increased. This implied that communicating mathematics across the Internet was problematic and so XSL helped to alleviate this problem by reducing file sizes and enabling mathematicians to transport notations and equations across the Internet.

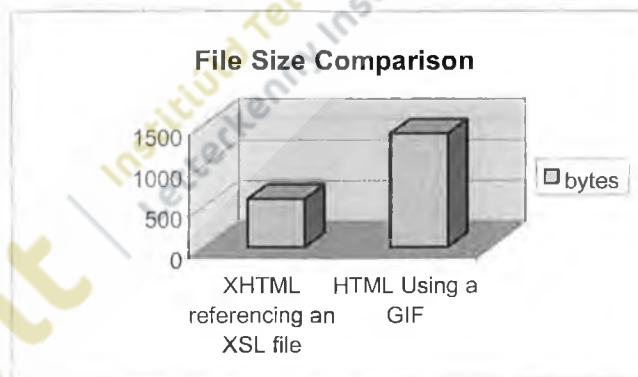


Figure 2 File Size Comparisons

If a mathematician wanted to render an equation on the Internet he would previously have needed to create a snapshot of the equation in either GIF or JPEG format. For the example used, a GIF was created (1,160 bytes in size). This was then embedded in a HTML page (232 bytes in size). This gave a total of 1,392 bytes of information. Alternatively, an XHTML file could be created (591 bytes in size) which could reference an XSL file from a URL and display the same notation in the form of markup. The difference in file size for transport across a network is substantial as can be seen in Figure 2.

It is clear that if a web site were to incorporate several GIFs and/or JPEGs the file sizes would increase exponentially and thus download times would be considerably affected. A bitmapped graphic contains a list of colours of individual pixels in a normally rectangular area [4]. Examples of these graphics include GIF, JPEG and Portable Network Graphics (PNG) images used on most Web pages. If a bitmapped graphic is 6 inches by 6 inches and

has a resolution of 72 pixels per inch, then it contains $72 \times 6 \times 72 \times 6$ pixels, that is, 186,624 pixels. If the image is stored in 24-bit colour, then each pixel occupies 3 bytes, therefore the image uses 4,478,976 bits (which is approximately 546.75KB^2 of memory). It is possible for the actual file to incorporate a variety of lossy and nonlossy compression algorithms to reduce the size. However, the calculations above prove how bitmapped images can increase in size very quickly. As a result of these large files, many Web pages with lots of pictures are so slow to load. With regard to rendering mathematics on the Web, the bandwidth issue can be reduced by incorporating XSL.

5 Conclusion

Communicating data is an ever increasingly important aspect of computing. Therefore, any technology whose objective it is to improve bandwidth capabilities across a network can only be considered a good thing. XSL and XSLT exist as a result of XML and XML's portability implies that these technologies could work on most platforms. Its primary drawback is the lack of support that currently exists. Nonetheless, the advantages that it does offer provides improved network capabilities for Internet users and developers.

References

- 1 Cagle, Kurt et al. "Professional XSL". Wrox Press Ltd. 2001. UK. ISBN: 1-861003-57-9.
- 2 Addey, Dave, et al. "Practical XML for the Web". Glasshaus. 2002. ISBN: 1-90415-1086. Excerpts Available Online at: <http://www.webreference.com/xml/resources/books/practicalxml/chapter5/>
- 3 BizTalk. Product Overview. Available Online at: <http://www.microsoft.com/biztalk/evaluation/overview/biztalkserver.asp>
- 4 Harold, Elliotte Rusty. "XML Bible". 2001. 2nd Edition. Hungry Minds Inc. USA. ISBN: 0-7645-4760-7.

² Calculation carried out in order to determine the Kilobytes of memory in use by the graphic: $(4,478,976 / 1024) / 8$

Published

**Proceedings of The Irish Signals and Systems
Conference 2003.**

July 2003.

lyit

Institiúid Teicneolaíochta Leictir Ceannainn
Letterkenny Institute of Technology

Representing Human-Computer Dialogues Using VoiceXML

Nigel Mc Kelvey*, Ruth Lennon M.Sc. MIEEE*, Thomas Dowling*.

**Computing Department, Letterkenny Institute of Technology, Port Road, Letterkenny, Co. Donegal, IRELAND.*

E-mail: nigel_mck@email.com*

Phone: 00353 87 6636014

Fax: 00353 74 9164107

Abstract -- A project is currently underway at the Letterkenny Institute of Technology in which research is being conducted in the field of Extensible Markup Language (XML). There are many different applications of XML but for the purposes of this paper only the Voice Extensible Markup Language (VoiceXML) will be discussed. This paper will analyse the structure of VoiceXML, its relation to XML and how this application can be implemented and utilised. In order to fully understand the structure of VoiceXML, it is necessary to have a brief introduction to XML. XML is a markup language for documents incorporating structured information. Structured information contains both content (words and pictures) and an indication of what function that content plays. XML provides an environment for the integration of unrelated forms of data in a platform that is independent. This data can include varying types of content. That is, content in a section heading has a different meaning to content in a footnote and so forth. Most documents have some structure. XML provides a standard format for the description of data thus allowing applications to retrieve data from a variety of web sites and subsequently to assimilate the data. The XML specification defines a standard way to add markup to documents. This document will outline the foundations of VoiceXML and the technical capabilities that it possesses. It will also highlight the advantages and disadvantages of incorporating this language into a Web application.

Keywords – VoiceXML, XML, IVR, SALT, TTS, Mark-up

I INTRODUCTION

Speech recognition technology has become an essential building block for a new set of next generation-enhanced services. VoiceXML is designed for creating audio dialogues that feature synthesised speech, digitised audio, recognition of spoken and Dual Tone Multi-Frequency (DTMF)¹ key input, recording of spoken input, telephony, and mixed-initiative conversations. When VoiceXML was created there were many aims of the venture but the primary goal of this language is to bring the advantages of web-based development and content delivery to interactive voice response applications [1]. With forecasts of 600 million PCs and two billion networked handheld devices by 2003, the IT environment is preparing to change to accommodate a new conversational user interface [2]. As VoiceXML is based on XML, it is necessary at this juncture to briefly discuss XML and its foundations.

XML is relatively new to the market therefore support for its features is somewhat limited. At present (2003) Microsoft's Internet Explorer 5 provides complete support for the Document Object Model (DOM) and Extensible Stylesheet Language (XSL). However, it does not provide complete support for Cascading Style Sheets (CSS), which is arguably the best means of displaying data in a browser. Netscape 6 (release 1) is similar, as it does not support the Extensible Stylesheet Language Transformation (XSLT). By using XSLT it is possible to turn an XML document into HTML, PDF or another form of XML. This paper will discuss how XML can be used to create other applications of the language such as VoiceXML. Figure 1 shows how a typical XML system operates.

An XML-enabled browser, for example, can offer improved querying and search capabilities for Web sites incorporating XML. The XML Document and XML Document Type Definition (DTD) pass their information to the parser. The parser is a program that can read XML syntax. By using a parser the finished application will never have to look at the XML for the information, thus speeding up the process of task execution and information display.

¹ DTMF are the tones used in telephones for tone dialling.

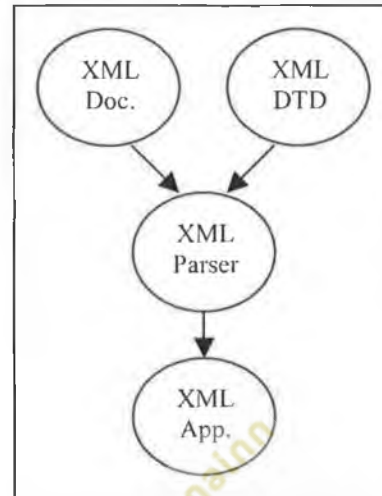


Figure 1: An XML System [3]

XML has facilitated the launch of various applications aimed at speeding up the process of creating and downloading graphics on the Web and providing voice recognition Computer Based Training (CBT) facilities.

a) Benefits of Incorporating VoiceXML

Much of what can be accomplished on the web, can also be attained using voice applications, and in many cases existing web content can be adapted for voice browsing. Similar to HTML, VoiceXML integrates well with JavaScript and Java [4]. At present (October 2002) it is quite difficult to embed a VoiceXML file into a HTML document. However a new specification has been released (in 2002) called Speech Application Language Tags 1.0 (SALT). SALT is a small set of XML elements, with associated attributes and DOM object properties, events and methods, which may be used in conjunction with a source markup document to apply a speech interface to the source page [5]. The benefits of developing applications through VoiceXML include [4]:

- Delivering web content and services through the telephone.
- Leveraging existing Internet infrastructures to facilitate this technology.
- Portability across implementation platforms.
- Decreases the level of expertise required to create voice applications.

- Enables rapid voice application development, similar to HTML for the web.

II VOICE EXTENSIBLE MARKUP LANGUAGE

VoiceXML is a Web-based markup language for representing human-computer dialogues, similar to HTML [1]. The VoiceXML language is based on the World Wide Web Consortium (W3C) XML standard. While HTML assumes a graphical web browser, with display, keyboard, and mouse, VoiceXML assumes a voice browser² with audio input and output. A typical VoiceXML voice browser runs on a specialised voice server that is connected to both the public switched telephone network and to the Internet as can be seen in Figure 2. These voice servers and interpreters extend the power of the web to telephones worldwide. VoiceXML is currently being used by The Weather Channel and CBS Marketwatch.com to deliver audible content and interpret spoken input [6].

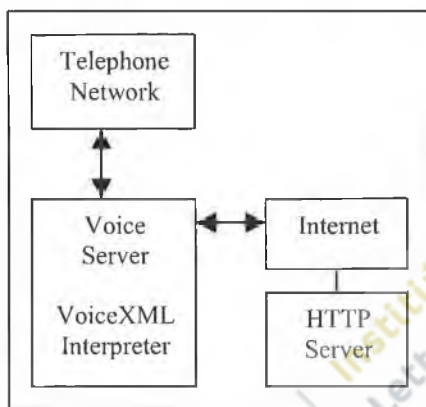


Figure 2: A typical VoiceXML voice browser [7]

As with all XML application interoperability is an objective of VoiceXML. VoiceXML protects application developers from issues such as multi-programming and platform specific APIs [9]. The abstraction attained provides improved service portability with platforms such as Interactive Voice Response (IVR³). Several organisations, including AT&T, are now taking advantage of this interoperability.

² Voice browsers allow people to access the Web using speech synthesis, pre-recorded audio, and speech recognition.

³ An IVR system lets the user phone up and "talk" to a computer system.

Using VXML, the user interacts with a voice browser through audio output that is either pre-recorded or computer-synthesised and submitting audio input through the user's natural speaking voice via a microphone or through a keypad, such as a telephone. VoiceXML technology enables application interface in the form of dialogues. Input is produced through speech recognition of end-user's voice and output is produced via text-to-speech (TTS⁴) technology such as Elan TTS.

Each VoiceXML application has a document specifying every interaction dialogue that a VoiceXML interpreter manages [8]. The input provided by the user effects dialogue interpretation, and the system gathers the data into requests that it subsequently submits to a document server. The document server can then reply with another VoiceXML document enabling continued user communications through dialogues [8]. Features of the dialogue include: the compilation of input, creation of audio output, management of asynchronous events and continuation of the dialogue [9]. VoiceXML supports the following input forms:

- Audio recording,
- Automatic speech recognition and
- Touch tone.

AT&T, IBM, Lucent Technologies, and Motorola joined to develop the VXML Forum in a joint effort to endorse the emerging technology [10]. The Forum produced VXML 1.0 as the initial version of VXML. The VXML Forum strived to create an open VXML specification for standardisation and to instruct the industry about the need for a standard voice markup language. The Forum also wanted to attract industry support and to encourage industry-wide use of the resulting standard to create inventive content and service applications [1].

Because it is an XML-based definition with an HTML-like appearance, VXML would be relatively easy to learn for experienced Web programmers and would be easily processed by tools that support desktop development of VXML Web applications.

In order to fully understand how these languages operate it is necessary to depict diagrammatically how they interact in a given system. Figure 3 shows the voice markup languages from a systems-level perspective.

⁴ TTS is the process, which converts text into voice.

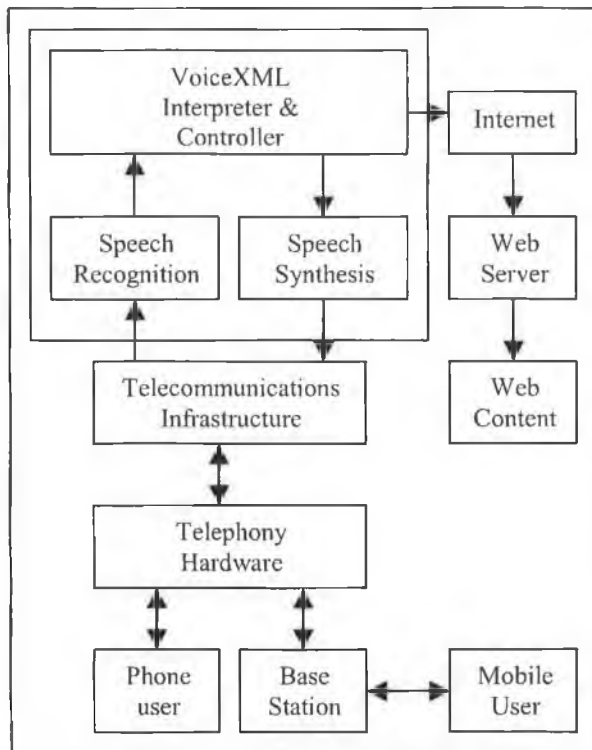


Figure 3: Telephone Interaction with a VoiceXML Interpreter [11]

The user interacts with the system by relating a message via a telephone. The message is then recognised by the speech recognition software and is forwarded to the Voice Interpreter. If the grammar is recognised, a synthesised response is given to the user or alternatively sent through the Internet to be delivered as Web content in a browser. In Code Listing 1 an example of VoiceXML code is given that requires the user to give an audio response when prompted.

The top-level element is `<vxml>`, which is mainly a container for dialogues. There are two types of dialogues: forms and menus. Forms present information and gather input whereas menus offer choices of what to do next.

In Code Listing 1 an audio file "*ListenCarefully.au*" is played to the user asking them to listen to the instructions carefully. Audio is played from a VoiceXML page by referencing the file using an `<audio>` tag, in much the same way as placing a reference to a streaming audio file in a web page.

```

<?xml version="1.0" encoding="iso-8859-1"?>
<vxml version="1.0">
<form>
<record name="recording">
<prompt>
<audio src="ListenCarefully.au" />
Please leave your message
and then press any key on the
keypad, or say "end of recording", to stop.
</prompt>
<grammar>end of recording</grammar>
</record>
<filled><prompt>
You said
<value expr="recording" />
</prompt></filled>
</form>
</vxml>
  
```

Code Listing 1: VoiceXML Code

The program then prompts the user to "*Please leave your message...*". The prompt element controls the output of synthesised speech and prerecorded audio. After the prompt has stopped the program will record everything that the user says until they have said the phrase, "*end of recording*" or have hit a key on the keypad. This 'phrase' is a pre-defined grammar, which the program recognises in much the same way, as a Java program would recognise a variable assigned a value. The recording is stored in the field item variable, which can be played back or submitted to a server. The program will then conclude by playing back the user input. The program as given in Code Listing 1 has been written and tested using the IBM WebSphere Voice Toolkit.

III RESEARCH

In order to test a VoiceXML system developed as part of research efficiently, several users were required to use the system. These individuals (4 females and 3 males) all had various accents (i.e. Northern Irish, Southern Irish and Russian, all speaking in English). Tests involved participants relaying their name as input to the VoiceXML system followed with the sentence "end of recording" or hitting a button on the keyboard to terminate the recording.

A voice driven menu was provided as part of the system which would require the user to make a tutorial selection such as, "SMIL, SVG or VoiceXML" etc. When the selection was made successfully, the user needed to enter in a credit card

number using either their voice or by using the keypad. The number relayed to the system by the user was verified using Java Script. A snippet of this code is given in Code Listing 2. Table 1 also shows some of the sample credit card numbers that are available. This was necessary in order to check the number entered by the user for length and format.

Credit Card Type	Format
Visa	4111111111111111
Master Card	5500000000000004
American Express	340000000000009
Diner's Club	30000000000004
Carte Blanche	30000000000004
Discover	6011000000000004

Table 1: Sample Credit Card Numbers

It was necessary to investigate if VoiceXML could interact with another language. Therefore, Java Script was used for research purposes.

```

<script><![CDATA[
function validCardType(CNumber) {
var cardLengths = new Array ( 'v', 13, 'v', 16,
'm', 16, 'a', 15, 'c', 14, 'd', 16);
var cardDigits = new Array ( 'v', '4', 'm', '51',
'm', '52', 'm', '53', 'm', '54', 'm', '55', 'a', '34', 'a',
'37', 'c', '300', 'c', '301', 'c', '302', 'c', '303', 'c', '304',
'c', '305', 'c', '36', 'c', '38', 'd', '6011');
var validCard = false;
var correctLength = false;
.....

function GetCCardName(CardType) {

switch(CardType){
case "v":
return "Visa";
break;
case "m":
return "Master Card";
break;
.....
}
}
.....
]]></script>

```

Code Listing 2: Java Script Snippet

Testing revealed a time delay anomaly. If a user were to relay the digits quickly a problem was not encountered, however if the user delayed, then the system would return to the start of the menu again or

terminate. This was resolved by increasing the delay allocated to each digit entered. In other words, the user was given more time to relay each digit without the system 'giving up' or 'timing out'. It was discovered that accents appeared not to have a significant impact on the overall functionality of the system.

VoiceXML enables relatively natural dialogues by allowing words to be given a certain articulation outside of <form>. For example, certain names may have a completely different pronunciation from what would be imagined judging by the spelling.

In order to test this theory a section of code was incorporated that would 'welcome' a user named 'Eilish' to the system. Before any modifications were made to the code, the system was run but the pronunciation was incorrect. However, it was possible to specify in the grammar how this name should be pronounced. Code Listing 3 shows how a word can have its pronunciation altered inside tags declared outside of the <form> tag. The <form> is VoiceXML's basic dialogue unit, which describes a set of inputs (<fields>) required from the user to carry out a transaction between the user agent (browser) and a server.

It was concluded that VoiceXML has numerous advantages to offer but a significant disadvantage would be that security issues might arise as a result of spoken dialogues being used to relay information.

```

<ibmlexicon>
  <word spelling="Eilish"
    pronunciation="e&#618;&#618;l.&#616;&#643;" />
</ibmlexicon>

```

Code Listing 3: Altering The Pronunciation of a Word

IV FUTURE WORK

In order to fully investigate the potential offered by VoiceXML a multi-modal application could be created that should endeavour to alleviate some of the problems encountered by individuals with disabilities on the Internet. A multi-modal system is one where a user can interact with Web and voice content at the same time. SALT tags can help bring this about. An example of a multi-modal system would be an application containing a map to a certain area that

provides alternative routes to a location on the map. The application could display several different routes to a specific location. The user would then be asked by the system to choose the route that they desire. The user might select by issuing the command, "Route A". From here more specific testing could take place that would investigate the capabilities of a voice driven Internet.

V CONCLUSIONS

From the research carried out it can be concluded that XML and VoiceXML together are working toward advancing the capabilities of the Internet. Technologies created through XML are striving to provide improved networking capabilities on the Web, which can be advantageous to general users as well as professionals. With faster downloads, improved capabilities and better integration of technologies, VoiceXML is fast becoming an asset to CBT and web-based services in general. It will help deliver voice services from the high-mobility worker on a mobile phone calling the company intranet to get information on a sales prospect to a person calling to get a weather report before going on holidays.

Computer users today (2003) need dependable, convenient access to the Internet wherever they are. The IT environment is preparing to change to accommodate a new conversational user interface as predictions of 600 million PCs and two billion networked handheld devices by 2003 were forecast [2]. With the arrival of speech recognition technology, the complexity and confusion of dealing with several interfaces is cut down.

VoiceXML protects application developers from issues such multi-programming and platform specific APIs [9]. The abstraction attained provides improved service portability with platforms such as IVR. Organisations such as AT&T, are taking advantage of this interoperability. It is the author's opinion that this technology will further expand within the near future.

REFERENCES

- [1] IEEE. "Industry Standards and Technology Organisation." 2001. Online. Internet. [December 2002]. Available WWW: <http://www.voicexml.org/tutorials>
- [2] Derouault, Anne-Marie. "The Future Of Speech Recognition". 7th January 2000. Online. Internet. [October 2002]. IBM Speech Systems, Document Number 05918. Available WWW: <http://www.advisor.com/Articles.nsf/ID/OA000107.DERO01>.
- [3] Ceponkus, Alex and Faraz Hoodbhoy, John Wiley & Sons, Inc., USA, July 1, 1999.
- [4] Penumaka, Srinivas. "VoiceXML: An Emerging Standard For Creating Voice Applications". Online. Internet. [December 2002]. Available WWW: http://www.sonify.org/tutorials/other/voicexml_intro/.
- [5] Speech Application Language Tags (SALT) 1.0 Specification, Cisco Systems Inc., Comverse Inc., Intel Corporation, Microsoft Corporation, Philips Electronics N.V., SpeechWorks International Inc. 15th July 2002;
- [6] "Westech's Virtual Job Fair & High Technology Careers." 1999. Online. Internet. [October 2001]. Available WWW: <http://www.highcareers.com/doc499>.
- [7] Martin, Didier. "Adapting Content for VoiceXML". 23rd August 2000. Online. Internet. [December 2002]. Available WWW: <http://www.xml.com/pub/a/2000/08/23/didier/index.html>.
- [8] "Is Speech Recognition Becoming Mainstream?", *IEEE Computer*, Vol 35, No. 4, April 2002, pp. 58-66.
- [9] "The Promise of a Voice-Enabled Web", *IEEE Computer*, Vol 33, No 8, August 2000, pp. 104-106.
- [10] Lucent Technologies, Bell Labs Innovations. 2nd March 1999. Online. Internet. [December 2000]. Available WWW: <http://www.lucent.com/press/0399/990302.bla.html>.
- [11] Houlding, David. "VoiceXML and The Voice-Driven Internet. A more natural interface to the Internet." Dr. Dobb's Journal. April 2001. Online. Internet. [June 2001]. Available WWW: <http://www.ddj.com/documents/s=868/ddj0104g/0104gfl.htm>.

Published

**Proceedings of The 32nd International Conference on
Computers and Industrial Engineering.**

August 2003.

lyit

Institiúid Teicnolaíochta Leitir Ceannainn
Letterkenny Institute of Technology

SMIL - PORTABILITY AND BROWSER SUPPORT ISSUES

Authors: *Nigel Mc Kelvey¹, Ruth Lennon M.Sc. MIEEE, Thomas Dowling. Letterkenny Institute of Technology*, Port Road, Letterkenny, Co. Donegal, Ireland. E-mail: nigel_mck@email.com

ABSTRACT

Synchronised Multimedia Integration Language (SMIL) is a markup language designed to be learnt and installed easily on Web sites. Problems exist concerning the coordination and display of multimedia on Web sites. Incorporating SMIL with a single time line for all media elements on a page aids the coordination and synchronisation of displays. Embedding low bandwidth data (e.g. text) in high bandwidth data (e.g. video) solely for the amalgamation can be eliminated with SMIL. One presentation however, could require up to five different file types, thus increasing complexity. An efficiently streamed SMIL presentation requires the files' relationships to be arranged and maintained, even whilst publishing. When changes are made to a presentation, those changes must often be reflected in multiple files. This places limitations on the portability of SMIL presentations. Additional topics documented in this paper include: SMIL portability across browsers, a number of SMIL capabilities/limitations and Web site advancements.

KEY WORDS

XML, SMIL, Browsers, Multimedia, Bandwidth

1 INTRODUCTION

The Extensible Markup Language (XML) was created in order to advance the functionality of the Web by enabling more flexible and adaptable information identification to be made available to users.

The primary reason for it being called eXtensible is because it is not a fixed format, like Hyper Text Markup Language (HTML). HTML is a single, predefined markup language. Alternatively, XML can be considered a 'metalanguage' (i.e. a language that describes other languages). XML also enables a developer to design a customised markup language for various documents. One of the main reasons why XML can do this is because it is written in Standard Generalised Markup Language (SGML). SGML is the international standard 'metalanguage' for text markup systems (ISO 8879).

¹ Author for correspondence

As a result of XML's popularity, many applications of XML have been developed such as Scalable Vector Graphics (SVG), Vector Markup Language (VML), Mathematical Markup Language (MathML), Voice Extensible Markup Language (VoiceXML) and Synchronised Multimedia Integration Language (SMIL) to name but a few.

2 SYNCHRONISED MULTIMEDIA INTEGRATION LANGUAGE (SMIL)

SMIL is used to express the behaviour and layout of multimedia presentations giving them some additional semantic significance (Moreno, 2002). SMIL is a tool used for building synchronous, streaming multimedia presentations that integrate audio, video, images, and text. The most popular SMIL browser is the Real Audio G2 player.

In a SMIL presentation, all of the media elements including images, audio clips, video clips, animations, and formatted text are referenced from the SMIL file. This is similar to the way a HTML page references its images, applets, and other elements. One of the first commercial SMIL players to emerge on the market was RealNetworks' "RealPlayer G2". Earlier versions of RealPlayer played only RealNetworks' audio and video file formats. However G2 includes support for many other media types such as WAV, AVI, JPEG, MPEG, and various others as can be seen in Table 1 (Marshall, 2001):

Media	Tag	G2	GriNS	Soja
GIF	Img	Yes	Yes	Yes
JPEG	Img	Yes	Yes	Yes
MS Wav	Audio	Yes	Yes	No
Sun Audio	Audio	Yes	Yes	Yes
Sun Audio Zipped	Audio	No	No	Yes
MP3	Audio	Yes	No	No
Plain Text	Text	Yes	Yes	Yes
Real Text	Textstream	Yes	No	No
Real Movie	Video	Yes	No	No
AVI	Video	Yes	Yes	No
MPEG	Video	Yes	Yes	No
MOV	Video	Yes	No	No

Table 1 *Media support in SMIL*

Displaying various media elements can be further demonstrated by viewing Code Listing 1, which incorporated several media elements into one presentation:

```
<smil>
<head><layout>
<root-layout width="431" height="334" background-color="#000066" />
<region id="pic_pos" width="110" height="48" left="26" top="14" />
<region id="text_pos" width="220" height="48" left="126" top="14" />
<region id="vid_pos" width="110" height="48" left="26" top="62" />
</layout></head>
<body>

<video src="video1.avi" region="vid_pos" begin="1.2s" fit="slice" />
<text src="text1.rt" region="text_pos" begin="1.2s" fit="slice" />
</body>
</smil>
```

Code Listing 1 *Various Media Elements in a SMIL Presentation*

The difficulties that are apparent between these elements can be expressed in three various ways: time-based, object-based or an amalgamation of both. It is imperative that when a person speaks their lip movements relate exactly to the sound being produced. Poor synchronisation is particularly evident if the sound arrives before the vision.

SMIL can be used to stream media elements over low bandwidths and it can also be used to stream media using protocols such as Hyper Text Transfer Protocol (HTTP) (Goose, 2002). This implies that SMIL could be utilised behind firewalls. It is also possible for a media player to be embedded within a Web page and incorporate controls (such as 'stop', 'play', 'pause' etc). This adds additional functionality to the Web as SMIL can subsequently be integrated with browser technologies already in existence. Code Listing 2 outlines the code that could be used to embed a SMIL presentation into a Web page:

```
<object id="media"
  classid="CLSID:CFCDA03-8BE4-11CF-B84B-0020AFBBCCFA"
  width="550" height="410">
  <param name="src" value="SMIL_Presentation_Name.smil">
  <param name="console" value="Clip1">
  <param name="controls" value="ImageWindow">
  <param name="AutoStart" value="TRUE">
  <embed controls="ImageWindow" console="Clip1"
  type="audio/x-pn-realaudio-plugin"
  src=" SMIL_Presentation_Name.smil"
  width="250" height="190" autostart="true">
  </embed>
</object>
```

Code Listing 2 *Embedding A SMIL Presentation in a Web page*

To use a Netscape plug-in, the <embed> tag is required, and with Internet Explorer the <param> tag is required. Code Listing 1 was implemented successfully using Internet Explorer 5.5. The 'type' attribute lets the browser know which plug-in to use in order to play the specified media. The 'src' attribute tells the plug-in what media to play and where it resides and the 'autostart' attribute is required to determine whether the clip starts playing automatically or not.

2.1 BANDWIDTH IMPORTANCE

It is important to note that the file formats of media that can be referenced in the , <text>, <video>, and other tags are implementation-dependent (Bouthillier, 1998). In other words, while SMIL defines an image content tag, a particular SMIL player may or may not support every image format (refer to Table 1).

SMIL presentations can be comprised of multiple elements and each of these elements requires some amount of bandwidth to be delivered to the receiving media player. The total bandwidth that all of the elements require for a presentation that plays simultaneously should not exceed the total bandwidth available to the target viewer (Boston). If this factor is not taken into consideration by the developer then limitations could be put on the portability of SMIL presentations.

It is also necessary for certain clips (such as images) to remain within the bandwidth boundaries established by their respective files (Boston). This can be further highlighted with an example. If a Real Pix file specifies a bitrate of 10, 000 (10K), then images in the file should also remain that size in order to prevent the presentation from crashing or timing out.

Streaming media can require more bandwidth in addition to computing resources. Streaming media requires the use of bandwidth based on the duration and quality of the media being delivered

(Confluent Tech.). Video requires more bandwidth than audio and a full screen video requires more bandwidth than partial screen video. When combining other media elements into a multimedia presentation, the additional media consumes further bandwidth.

Table 2 outlines some of the media components that may be incorporated and some suggested bandwidths (Boston):

Media	Description	Bandwidth
Audio	.rm, .au	34kbps +
Image	.gif	12kbps
Text	.txt	< 1kbps
Video	.avi	256kbps approx.

Table 2 *Media Components/Suggested Bandwidths*

It is possible to add a variety of different transition effects between still images. RealPlayer G2 renders the effects at playback time and as a result the affect on the download is marginal (Heid, 2000).

2.2 WEB SITE ADVANCEMENTS – CATERING FOR THE DISABLED

Individuals with cognitive or learning disabilities, such as dyslexia, need a more general solution to accessing information on the Internet, such as providing a consistent design and using simplified language on Web sites. It is important for a Web developer to incorporate a similar layout and design for each page, so a person with a cognitive disability can learn to navigate through a Web site a little easier. People with cognitive or learning disabilities can also benefit from redundant input (Weiser).

Redundant input implies that the developer provides both an audio file and a transcript of a video or spoken dialogue (Weiser). Viewing the text and hearing it read aloud simultaneously, enables a person with cognitive and learning disabilities to take advantage of both auditory and visual skills to understand the material better. SMIL can help offer these capabilities to users.

Providing a Web site that caters for people with hearing difficulties requires a similar approach to that of providing accessibility to those with visual impairments. Where information is relayed using audio techniques, a textual alternative is also required. SMIL could be advantageous in this sense as information can be relayed on a given topic through spoken dialogue and can also be accompanied by a 'slide' that presents the same information textually.

Figure 1 depicts the individual multimedia components that make up a SMIL presentation:

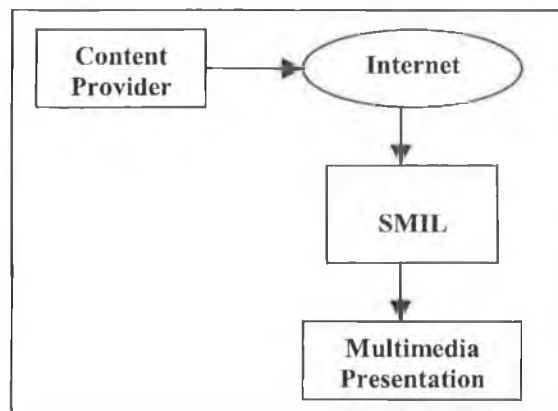


Figure 1 *Multimedia Components in a SMIL Presentation*

SMIL presentations can play in a browser with a SMIL plug-in or in a standalone player such as RealOne or QuickTime that reside on consumer devices and are independent of browsers (Coyle, 2003). The SMIL file declaratively describes how the content will be displayed. A browser that has a plug-in that recognises a SMIL file will be able to locate the media files and run the presentation. The user will then be able to avail of a synchronised multimedia presentation.

2.3 SCALABLE PROFILES

A scalable profile enables users of varying programming abilities with varying technologies to create SMIL documents that will act intelligently and which are tailored to the capabilities of the target devices. Conformance to a SMIL Basic profile provides a basis for interoperability guarantees (Morrison, 1999). Some QuickTime SMIL browsers cannot decode certain media objects thus network problems can arise.

A SMIL presentation can consist of several components that are accessible via Uniform Resource Locator (URLs) (e.g. files stored on a Web server). These components can have many varying media types such as audio or video etc. The times that these components begin and end at can be specified in the markup and the interface can be improved by incorporating various buttons that allow the user to play the presentation to their own specifications (i.e. more slowly, paused, fast forward etc.).

3 CONCLUSION

To conclude, SMIL has many advantages that are helping to enhance the availability of various media elements on the Internet. It is enabling these elements to be streamed more effectively which is making information more readily available and to a wider audience. Considering that the media is streamed using a markup, this implies that the information is presented more effectively.

SMIL is regarded as a language that can be easily learnt, however, if bandwidth issues are not considered carefully, then the quality of the playback can be considerably reduced. It is therefore the author's opinion, that more specific guidelines should be made available in order to accommodate developers of varying skills.

The SMIL Basic profile helps guarantee interoperability for the SMIL presentation. This helps make the applications more portable. In order to make Web sites more accessible to individuals with disabilities, then redundant input is required. This requires the developer to provide both an audio file and a transcript of a video or spoken dialogue to the user. SMIL can facilitate this as various media elements can be streamed into one application.

SMIL may not necessarily replace the technologies that already exist (e.g. animated Graphics Interchange Format (GIFs), Shockwave or Java Applets etc.) enabling multimedia to be placed on the Web, but it can be considered a useful alternative.

REFERENCES

- Moreno, 2002 Moreno, P.J., Van Thong, J.M., Logan, B. & Jones, G.J.F. (2002). From Multimedia Retrieval to Knowledge Management. *IEEE, Computer*. Volume 35, Number 4. (pp. 58-66).
- Marshall, 2001 Marshall, D., Let us begin to SMIL-SMIL Authoring. (2001). Available Online at: <http://www.cs.cf.ac.uk/Dave/Multimedia/node99.html>
- Goose, 2002 Goose, S., Kodlahalli, S., Pechter, W. & Hjelsvold, R. (2002). Streaming Speech: A Framework for Generating and Streaming 3D Text-To-Speech and Audio Presentations to Wireless PDAs as Specified Using Extensions to SMIL.

- Proceedings of the 11th International Conference on World Wide Web.* International World Wide Web Conference. ACM Press. (pp. 37-44).
- Bouthillier, 1998 Bouthillier, L., Synchronised Multimedia On The Web. (1998). *A New W3C Format Is All Smiles.* Available Online at: <http://www.webtechniques.com/archives/1998/09/bouthillier/>
- Boston Creating Multimedia with SMIL. *Boston University.* Available online at: <http://www.bu.edu/wedcentral/learning/smil1/bandwidth.html>
- Confluent Tech. Maximizing Your Media with SMIL. *Confluent Technologies Inc.* Available Online at: <http://www.fluition.com/training/maximizingyourmedia.pdf>
- Heid, 2000 Heid, J., SMIL: Markup for Multimedia. (2000). *Create Bandwidth-Friendly Streaming Multimedia.* Available Online at: <http://www.creativepro.com/story/feature/3496.html>
- Weiser Weiser, M. An Embedded, Invisible, Every-Citizen Interface. *Xerox Palo Alto Research Centre. Position Papers, On Interface Specifics.* Available Online at: <http://www.nap.edu/readingroom/books/screen/11.html#nomad>
- Coyle, 2003 Coyle, F., XML in Higher Education. SMIL: Multimedia Rides the XML Wave. (2003). *Syllabus.* Volume 16, Number 8. (pp. 22-25). Available Online at: <http://www.syllabus.com/article.asp?id=7361>
- Morrison, 1999 Morrison, M., et al. (1999). *XML Unleashed - From Knowledge to Mastery.* SAMS. 1st Edition.

Published In The
The Irish Scientist Year Book 2003.
November 2003.

lyit

Institiúid Teicneolaíochta Leictir Ceannainn
Letterkenny Institute of Technology

– The Rejuvenator!

XML (eXtensible Markup Language) as an enabling technology is rapidly advancing many legacy programs into the 21st Century. The adaptability of XML has allowed for the development of ‘new languages’ with which to write specialised applications. Research being carried out at Letterkenny Institute of Technology (LYIT) has explored the capability of XML to rejuvenate dated applications and to access data stored in such ‘prehistoric’ applications. We are all interested in the Information Age but are sick of receiving too much information and not receiving relevant information. The application of XML to retrieving data from a variety of sources for amalgamation allows for the filtering of massive amounts of information to select only pertinent information. Indeed, the vast leaps in technology have often caused information stored in legacy databases to be overlooked. This problem is being corrected through the use of XML. The efficient application of XML with additional technologies, allows for the retrieval of data from a variety of sources including legacy databases for the purpose of finding specific information. Indeed databases such as Oracle and Access can be accessed and their data combined for search purposes through XML. In this way data stored in these older databases can be accessed and more efficiently filtered through XML.

This is merely the tip of the iceberg. The application of XML based languages can range from Graphic Information Systems (GISs) to Voice Driven Applications to assist the disabled. GIS applications can be used for navigation of roads, map enhancements, etc. In this the year that Ireland hosts the Special Olympics, it is especially fitting to look at the XML can play in enabling the use of computers by the disabled. XML is also finding increased popularity in the provision of timing functions in HTML pages. This enhances web pages by providing the ability to synchronise displays, graphics, sound etc. Research at LYIT has focused on the use of XML to rejuvenate legacy systems while outlining the strengths and weaknesses of this technology. Applications developed have proven the capabilities of XML in the aforementioned areas and have highlighted a number of very interesting points. XML technologies applied include VoiceXML, SMIL (Synchronised Multimedia Integration Language), VML (Vector Markup Language), XML, SVG (Scalable Vector Graphics), MathML, etc.

Authors: Nigel McKelvey, Ruth Lennon & Thomas Dowling: Computing
Department, Letterkenny Institute of Technology (LYIT).
Contact: Nigel McKelvey, 087 6636014. E-Mail: nigel_mck@email.com