INVESTIGATING THE EFFICACY OF XML AND STYLESHEETS TO RENDER ELECTRONIC COURSEWARE FOR MULTIPLE LEARNING STYLES

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE,

FACULTY OF SCIENCE

AT THE UNIVERSITY OF CAPE TOWN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF PHILOSOPHY (IN INFORMATION TECHNOLOGY)

BY

MASHA DU TOIT

JANUARY 2007

SUPERVISED BY

HUSSEIN SULEMAN

Acknowledgments

Completing this project would not have been possible without the help and support of the following people. I would like to take this opportunity to thank them:

- My husband, Brendon Bussy, without whose endless patience, guidance and support I would never have survived this process.
- My supervisor, Hussein Suleman for providing timely support, much technical help and whose sane advice kept me on track.
- My colleagues and students for allowing me the time to focus on my studies.
- All those who participated in discussing and testing this project, often on very short notice during busy times.
- My family, in particular my mother and father, who were endlessly supportive, interested and concerned about the progress of this paper.

Abstract

The objective of this project was to test the efficacy of using Extensible Markup Language (XML) - in particular the DocBook 5.0b5 schema - and Extensible Stylesheet Language Transformation (XSLT) to render electronic courseware that can be dynamically re-formatted according to a student's individual learning style.

The text of a typical lesson was marked up in XML according to the DocBook schema, and several XSLT stylesheets were created to transform the XML document into different versions, each according to particular learning needs. These learning needs were drawn from the Felder-Silverman learning style model. The notes had links to trigger JavaScript functions that allowed the student to reformat the notes to produce different views of the lesson.

The dynamic notes were tested on twelve users who filled out a feedback questionnaire. Feedback was largely positive. It suggested that users were able to navigate according to their learning style. There were some usability issues caused by lack of compatibility of the program with some browsers. However, the user test is not the most critical part of the evaluation. It served to confirm that the notes were usable, but the analysis of the use of XSLT and DocBook is the key aspect of this project. It was found that XML, and in particular the DocBook schema, was a useful tool in these circumstances, being easy to learn, well supported and having the appropriate structure for a project of this type.

The use of XSLT on the other hand was not so straightforward. Learning a declarative language was a challenge, as was using XSLT to transform the notes as necessary for this project. A particular problem was the need to move content from one area of the document to another - to hide it in some cases and reveal it in others. The solution was not straightforward to achieve using XSLT, and does not take proper advantage of the strengths of this technology. The fact that the XSLT processor uses the DOM API, which necessitates the loading of the entire XML document into memory, is particularly problematic in this instance where the document is constantly transformed and re-transformed. The manner in which stylesheets are assigned, as well as the need to use DOM objects to edit the source tree, necessitated the use of JavaScript to create the necessary usability. These mechanisms introduced a limitation in terms of compatibility with browsers and caused the program to freeze on older machines. The problems with browser compatibility and the synchronous loading of data are not insurmountable, and can be overcome with the appropriate use of JavaScript and the use of asynchronous data retrieval as is made possible by the use of AJAX.

Contents

List of Acronyms

ADL

Advanced Distributive Learning

AJAX

Asynchronous JavaScript and XML

CAI

Computer Aided Instruction

CSS

Cascading Style Sheets

DOM API

Document Object Model Application Interface

DTD

Document Type Definition

HTML

Hyper Text Markup Language

IEEE

Institute of Electrical and Electronics Engineers

IMS

Information Management Systems

LMS

Learning Management System

LOM

Learning Object Metadata

NIST

National Institute for Standards and Technology

PDF

Portable Document Format

SCORM

Shareable Content Object Reference Model

SGML

Standard Generalised Markup Language

W3C

World Wide Web Consortium

WHURLE

Web-based Hierarchical Universal Reactive Learning Environment

XML

Extensible Markup Language

XSL

Extensible Stylesheet Language

XSL-FO

Extensible Stylesheet Language Formatting Object

XSLT

Extensible Stylesheet Language Transformation

XUL

XML based User interface Language

List of Figures

Chapter 1: Introduction

Computer Assisted Instruction is a vast and ever growing field. New technologies offer new possibilities. There is the Internet's promise of accessibility and the possibility of learner-driven, tailor made learning material offered by interactive media, to name but a few. New technologies offer solutions to old problems, but it remains to be seen how they perform in practice and what the most appropriate application is.

In the following section the motivation behind the research is discussed.

1.1 Motivation

A problem pertinent to all learning material, whether it is print or computer based, is the differing learning needs of individual students. Individuals have different "learning styles", and benefit from being presented with learning material that suits their learning style. However, to create duplicate sets of learning material to suit any possible combination of learning needs would be inefficient, time consuming, expensive, prone to error and difficult to update.

Information technology in the form of XML and XSLT gives us the tools to re-create a single source document in many different ways. Computer aided instruction offers alternative ways for students to navigate through material. How suitable are technologies such as XML and XSLT to encode and render notes for multiple learning styles?

The following section outlines the main objectives of this research.

1.2 Aims and Objectives

The aims of the research were:

- To investigate the use of XML and stylesheets to render electronic courseware for multiple learning styles.

The precise objectives of the research were:

1 To establish the efficacy of using Extensible Markup Language (XML) - in particular the DocBook 5.0b5 schema - for marking up learning material intended to be re-formatted by the student according to their individual learning style.

2 To establish the efficacy of using Extensible Stylesheet Language Transformation (XSLT) to transform the learning material in its DocBook XML form in such a way that it

can be re-formatted by a student to suit their individual learning style.

In the following section, the methodology used in the project in order to achieve the project objectives is described.

1.3 Methodology

In this project, a set of notes - referred to hereafter as the "dynamic notes"- was created. The text of a typical lesson was marked up in XML according to the DocBook schema, and several XSLT stylesheets were created that transformed the content of the XML into a number of different versions of the notes. Each of these versions addresses particular learning needs of individuals. These learning needs were based on the Felder-Silverman learning style model.

In the finished dynamic notes the student can - with the help of links that trigger various JavaScript functions - reformat the notes to produce different versions of the lesson. For example, a student might prefer to organise the notes so that all suggested activities are grouped at the end of the lesson, or alternatively, are placed throughout the body of the lesson. The notes were tested on a number of users who filled in a feedback questionnaire (Appendix B).

1.4 Assumptions

As discussed in the literature review section of this paper, there has been much research done on various learning style models and on the advantage to a student of having learning material presented according to their learning style. The assumption is made that the learning style model is valid, and that, if learning material is presented according to such a model, there will be a benefit to the student.

The next section outlines the scope and limitations of the project.

1.5 Scope and Limitations

This project looks at the area of Computer Aided Instruction as opposed to Computer Aided Assessment. The aim is to create notes that are usable, understandable and easily navigable according to an individual's learning style. The system does not assess the student's progress or learning style.

The dynamic notes are intended as a way for students to generate notes to be printed. Although there are interactive elements, and the user interacts with these on the computer to customise their version of the notes, ultimately the intention is that the student will print the

notes. This removes the need for complex navigation and paging systems, and to some extent limits the problems created by browser incompatibility.

1.6 Organisation of this Dissertation

This section provides a chapter by chapter summary of the dissertation:

Chapter 2- Background and Literature Review: this section gives an overview of the technologies used in this project, and presents an overview of some of the existing projects done in similar areas.

Chapter 3- Method, analysis and evaluation: in this section the process of the creation of the dynamic notes is described:

- Initial interview and survey process to establish the basic principles on which the notes should be based.
- The process of choosing categories according to the Felder-Silverman learning style model to prepare the notes for markup in XML.
- Mapping the chosen categories to the elements and attributes available in DocBook.
- The creation of the XSLT stylesheets.
- The use of DocBook and XSLT is evaluated.
- The feedback from the user test is presented and summarized.

Chapter 4 - Conclusion: summarises the findings, and discusses to what extent the original aims of the project were met. The implications of the problems encountered are discussed, and suggestions for future work made.

Chapter 2: Background and Literature Review

This section gives a summary of the various concepts and technologies encountered during this project. It is also an overview of some other projects that have been done in similar areas. It includes a description of Computer Aided Instruction and the various ideas and problems associated with it. It introduces learning style models and gives a quick synopsis of the learning style model used in this project: the Felder – Silverman model. The technologies of Extensible Markup Language (XML), the DocBook schema, Extensible Stylesheet Language (XSL) and Extensible Stylesheet Transformation Language (XSLT) are described in some detail. Standardisation as applied to XML and Computer Aided Instruction in particular is examined.

2. 1 Computer Aided Instruction

Computer Aided Instruction (CAI) is the practice of using a software system to support the learning of one or more students. CAI is generally believed to offer increased efficiency, improved accessibility and the possibility of cutting costs. Assumptions are made that computers benefit teaching and learning and can improve student academic performance. Research demonstrates that the way that computer technology is integrated into the learning environment and the context in which it is used is crucial in determining its effectiveness (Lunts, 2002).

This project deals with the presentation of information according to an individual's learning style. Some CAI systems assess the learning style of a student, keep track of their progress and test them. This project does not to look at these areas of Computer Assisted *Assessment.* Instead it will focus on Computer Aided *Instruction.*

One of the greatest benefits of using computer-based notes as opposed to printed notes or books is that these notes can be changed and updated at one location without the need to re-distribute the updated version.

2.1.1 Types of CAI

There are different types of CAI. Some act as a supplement to other methods of instruction. Some are designed to stand alone and are referred to as *primary* CAI. Primary CAI is often part of distance education courses, while *supplementary* CAI forms part of a larger plan, most probably in a classroom situation (Reeves, 1993). This project deals with supplementary

CAI. The content was derived from a classroom situation, and it was tested in such a situation.

2.1.2 Learner Control

Another important factor in effectiveness of CAI is that of the control of the sequence and direction of instruction. Who has this control: the system or the student? A CAI system might be weighted towards program control, in which the creators of the system pre-define the manner in which a student has access to content, or toward a system in which the student can control their own access and direction (Reeves, 1993). This is referred to as the amount of *learner control* in a system. Typically a CAI system will integrate these two approaches by choosing a balanced combination of the two.

2.2 Learning styles

Learning style is defined as characteristic strengths and preferences in the way a learner takes in and processes information (Felder, 2002).

There are many models of learning styles. Some examples are:

- The Myers Briggs Type Indicator that is derived from Carl Jung's theory of psychological types.
- The Herman Brain Dominance Instrument, which categorises individuals based on the functioning of their physical brain (Felder, 2002).

There seems to be little evidence that any one model of learning styles is more effective than any other. Felder himself (the co-originator of the widely used Felder – Silverman Learning Style model) writes that that it is "almost immaterial" which learning style model is chosen, as they are essentially identical (Felder, 2002).

2.2.1 Computer Aided Instruction and Learning Style

Many studies have been done on the effectiveness or otherwise of teaching an individual according to their learning style. Most have shown convincingly that if an individual is taught exclusively in a manner not matched to their learning style, they might not be able to perform to the best of their ability (Bajraktarevic et al, 2000).

It does not follow that an individual should be taught exclusively according to their dominant learning style. Felder makes the observation that "To be competent in any field, a person needs to work well in all learning style models" and that if a lecturer should teach exclusively

in a student's preferred mode, students may not develop the mental dexterity needed to excel in their chosen field (Felder, 2002). Teaching students about learning style models in itself may be beneficial to their ability to learn. As Felder says:

"Teaching students about learning styles helps them learn the course material because they become aware of their thinking processes "(Felder, 2002).

Few researchers seem to have considered the idea of developing an individual's mental dexterity by exposing them to alternative approaches to learning.

2.2.2 Existing projects on Computer Aided Instruction and Learning Style

Most of the studies I came across focussed not on an entire Learning Style model but on an aspect of them. A popular choice is the Felder-Silverman Learning Style Model, and the researchers often limit themselves to a particular part of the model – such as looking only at the global/sequential or visual/verbal types. This limitation seems to be influenced by the technology used – for example a CAI system based on hyperlinked text lends itself to exploring the global/sequential dichotomy but would not be particularly helpful in looking at the sensing/intuitive dimension (Lunts, 2002).

An example of such a study took place at the University of Southampton. Bajraktarevic (Bajraktarevic et al, 2000) conducted a study with a group of school children. The children were presented with study material that was matched or mismatched to their preferred learning style, and then tested. The hypothesis for the test scores were that students who took part in the matched sessions would score significantly higher than students who were given material not matched to their preferred learning style.
There was sufficient evidence that the students who studied the matched course material benefited and so scored higher in the test.
The other hypothesis that the study attempted to prove was that the browsing time of the matched sessions would be significantly less. In fact, they found that there was no significant difference in the time spent. The researchers speculated that a reason for this might be that material that is geared towards a student's learning style also has the ability to draw them in more successfully – to "grab their attention".

This seems a less than satisfactory answer, especially when reading Rothermell and Tropea's paper on methods on the use of "Computer-Assisted Learning to Promote Careful Reading and Logical Skills". They make the point that "The goal should be careful and attentive

reading regardless of speed" (Rothermel, 1989). How does one determine whether a student has read "carefully and attentively"? The time spent on the material does not seem to be a good indication.

2.2.3 The Felder – Silverman Learning Style Model

The learning style model that was used in this project is the Felder-Silverman model. According to this there are four "learning dimensions" – four aspects to the process of learning: Perception, Input, Processing and Understanding. This is a recent version of the model, which differs from the original as presented in the paper "Learning and Teaching Styles in Engineering Education" in 1988. Since that time, the "Organisation" learning dimension with its preferences for either "inductive" or "deductive" learning was removed in the belief that inductive presentation is the generally more effective, and the "input" dimension was changed from "visual/auditory" to "visual/verbal"(Felder, 2002).

As represented by the diagram in Figure 1, each learning dimension is presented as a scale with extreme preferences at either end. For example, the "Input" dimension is presented as a scale between "Very Visual" and "Very Verbal". A person's overall learning style is a combination of their scores on the scales of all of the four learning dimensions.

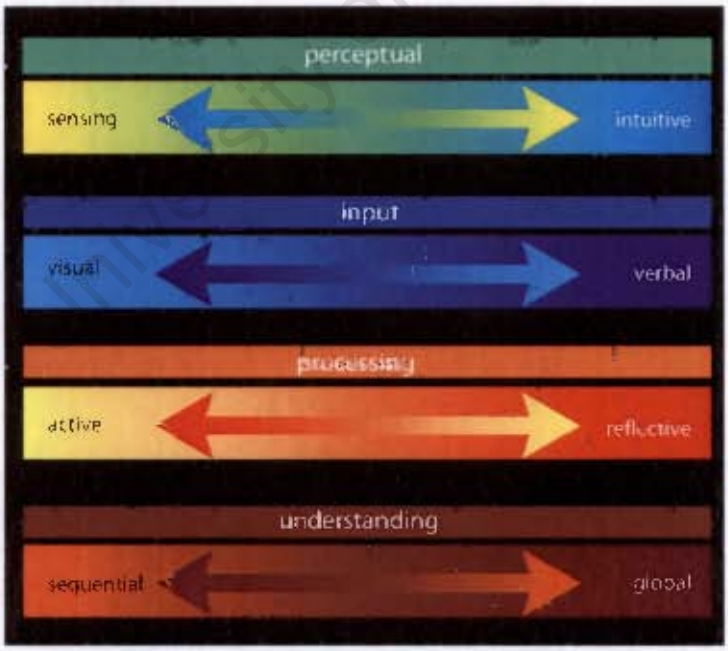

Figure 1: The four learning dimensions of the Felder-Silverman Model

2.2.4 The learner's needs according to the Felder-Silverman Model

Learning Dimension: Perceptual

Sensing: have a preference for real world analogies rather than symbols or theories. They need to know what the practical application of a lesson is before they feel comfortable absorbing information. They learn best by solving practical problems and repetitive exercises.

Intuitive: prefers to start by understanding the underlying theoretical basis of a lesson. They are comfortable with symbols and concepts and abstract ideas. They are likely to arrive at answers by using creative guesswork based on their understanding of basic theoretical concepts. They don't like repetitive work.

Learning Dimension: Processing

Active: need to learn while doing. They would experiment, discuss, explain and test rather than observe and think. They learn best by doing hands-on tasks and observing the consequences.

Reflective: are best able to gather information and then think about the possible consequences of what they have learnt. They will arrive at answers by reflecting on the new information and drawing conclusions.

Learning Dimension: Input

Visual: need to see diagrams, graphs, pictures, flow charts or timelines to help them understand and remember information.

Verbal: prefer to get information verbally – by reading or listening, and by discussing that information. They learn best when they are explaining that information to someone else.

Learning Dimension: Understanding

Sequential: Is most comfortable in being presented with information in small logical steps gradually forming a big picture. They are comfortable learning without understanding what the eventual application of the information might be. They can work with material without understanding it fully.

Global: need to know what they are working towards when learning. They need to see the context, the big picture and the relevance to their subject from the start. They struggle to

work with information that they don't understand completely, and benefit from being allowed to work out their own solutions to complex problems (Felder et al, 1988).

2.3 Markup Languages

2.3.1 Standard Generalised Markup Language (SGML)

To make this project a reality, a technology was needed to render the learning material in such a way that it could be re-formatted. A combination of XML and XSLT was chosen.

A markup language is a system which makes it possible to include information in a document, which contains "information about the information" in the document. Having "information about the information" makes it possible to process the data in a document in various ways. For example, it allows one to target and extract certain parts of the document, or specify that certain parts are displayed in various ways (Shepherd, 2001). There are a number of markup languages, based on the Standard Generalised Markup Language (SGML) developed by Charles Goldfarb, Ed Mosher and Ray Lorie in the early 1970's. SGML is a specification for developing markup languages (Anderson, 2004).

2.3.2 HTML

Out of the SGML specification emerged the very successful Hyper Text Markup Language (HTML), an example of a markup language that conveys details of presentation as well as some information on the meaning of the data. While some HTML "tags" provide semantic information on the content of the document, much of HTML tends to deal with the presentation details. For example some tags convey information on which parts of the document are paragraphs, or should be displayed in bold type – but you could not tell which part of the document represented the name of the author, or a telephone number (Shepherd, 2001).

2.3.3 Cascading Style Sheets

Cascading Style Sheets (CSS) were developed so that presentation details could be separated from the semantic markup of the document. All presentation rules are placed in a separate document called a "style sheet". Such a stylesheet could be embedded in a particular HTML document, or linked to it as an external file. While CSS stylesheets make it possible to separate the presentation details, they have limitations. With CSS, one can only define how existing elements are displayed. It cannot be used to add, change or remove elements of the document, and there are limits to the extent to which the document can be transformed (Addey et al, 2002).

2.3.4 XML

XML was developed as a markup language specification that could, amongst other goals:

- be usable over the Internet.
- be human as well as machine legible.
- make it simpler to write applications that process XML (Bray et al, 2006).

Like SGML, XML is not in itself a markup language, but is a specification for defining markup languages. XML is a set of rules for creating structured data that is machine readable. XML markup describes the structure and the meaning of the data. It is possible to create "author name" or "telephone" elements to convey information about the purpose or meaning of the data. This is what makes it possible for XML to be "machine readable". While it is human readable, no human action is necessary to extract the appropriate data (Shepherd 2001).

2.3.5 Well Formed XML

The rules of XML were established by the World Wide Web Consortium (W3C) as part of the XML version 1.0 specification. They include the requirements that:

- An XML document has only one root element that contains all the other elements and their content.
- All elements must have both opening and closing tags, and these must be case identical.
- Elements must be either sequential or properly nested with no overlap of start and end tags.
- XML processing instructions must be in lowercase. While elements may be in any case, they are case sensitive so the case used must be consistent.
- Attribute values must be quoted.
- Attribute values must be enclosed within single or double quotes (Bray, 2006).

These rules make it straightforward to create applications that generate XML markup, as well as applications that interpret XML markup. The rules are strictly enforced by the XML parser to ensure that the document is "well formed". The following is an example of XML that is not well formed:

```
<books>
  <TITLE>Hatchet</title
  <title id = 1>The Golden Shadow</Title)
  <title>Fire on the Deep</title>
  </Books >
  <books>
  <title>Hatchet
</books></title>
```

It does not have a single root element that contains all the child elements, some of the tags are not properly closed, it contains tags with differing case in the opening and closing tags, an attribute that is not in quotes, and improperly nested elements. It could instead be written like this:

```
<library>
   <books>
     <title>Hatchet</title>
     <title id = "1">The Golden Shadow</title>
     <title>Fire on the Deep</title>
   </books>
   <books>
     <title>Black Swan Green</title>
   <books>
</library>
```


2.3.6 Valid XML: Schemas

The structure that a particular XML document should take can be specified by an XML Schema – a document which declares a set of rules for how a specific XML document should be structured. Some parsers can compare an XML document to its schema to test whether it is "valid" according to the rules set out in that schema (Walsh, 2004). Some of the aspects that a schema would define can include:

- A list of elements that a document can include.
- A list of attributes that can be used.
- Specifications as to which elements can be nested in which, and the sequence in which they should be nested.
- Specifications as to what type of data specific elements or attributes can contain (Shepherd, 2001).

2.3.7 Customising XML

Despite the strict rules of syntax and structure, XML is highly customisable. An individual can create their own set of XML tags and their own schema to suit a particular situation. As long as an XML document is well formed - as long as it conforms to the syntax rules specified by the W3C as part of the XML version 1.0 specification - it can be interpreted by any application that is XML capable. XML is platform independent, license-free and widely supported (Addey et al, 2002).

Another strength of XML is that the details of presentation and style can be kept in separate style sheets. This makes it possible to generate a number of documents from the same XML – which differ in their presentation only. To update the presentation of these documents, it is only necessary to update the style sheets, and not the original XML document (Walsh, 2004).

2.3.8 XSL

Extensible Stylesheet Language (XSL) makes it possible to control how an XML document is presented. As XML deals with the syntactic markup, XSL deals with details of presentation and styling. In contrast to Cascading Style Sheets (CSS), which can only create rules for how a document should be presented, XSL can completely transform an XML document into a different markup structure. It was based on the Document Style Semantics and Specification Language (Addey et al, 2002).

There are two stages to this process of transformation: the structural transformation in which the elements are chosen and re-organised, and the formatting process in which the resulting elements are rendered in the appropriate format. So XSL is split into two markup languages:

- XSL Formatting Objects (XSL-FO) describes the presentation details of a document.

- XSL Transformations (XSLT) defines how to transform the XML document into another XML document.

A third part of this process emerged as it was found that the expression syntax being developed in XSLT to select parts of a document was similar to XPointer – which was being developed for linking one document to another. The two projects were joined to form the XPath expression language. XPath identifies particular parts of an XML input document to be processed, and performs calculations (Tennison, 2004).

2.3.9 XSLT

There are a number of ways to transform XML into other formats. Special purpose programs can be created with traditional programming methods using languages such as, for example, C++. A drawback of this approach is that if the format of the input or output changes, the special purpose program would have to be adapted.

With XSLT the developer uses a ready made general purpose program: the XSLT processor. If the XML needs to be transformed in ways that differ from the default ways set in the XSLT processor, the developer can customise XSLT stylesheets. An XSLT stylesheet is a set of rules that describe how a source XML document should be transformed into the result document (Tennison, 2004).

XSLT makes it possible to produce many different documents from a single XML document. These documents could be any of a number of formats: examples are HTML, Portable Document Format (PDF) or even another XML document. An XSLT stylesheet can specify how the information in the XML document should be displayed once it is transformed into its new format. It can re-organise the information in a specific order, add images, links or additional text and change the styling of the document. An XSLT stylesheet is an XML document. It has to conform to the same rules of syntax to be considered well – formed (Tennison, 2004).

2.3.10 The Transformation Process

When an XML document is transformed the processor typically creates two XML node-trees.

Referring to Figure 2: The XSLT processor creates "3", a source tree that is based on the node structure of the original XML document. It refers to the XSLT style sheet for information on how each part of source tree "3" should be transformed, and creates "5": a result tree that reflects whatever changes were made during the transformation process. This result tree contains Formatting Objects that control presentation aspects of the transformed document, which is interpreted by the Formatting Object Interpreter and so transformed into the new format (Addey et al, 2002).

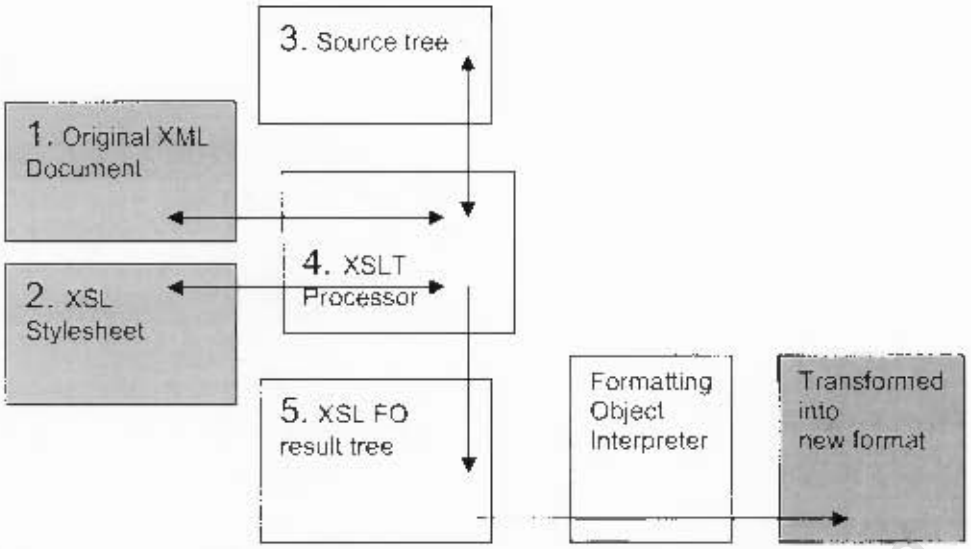

Figure 2: The transformation process of XML using XSLT

2.3.10 Transforming XML with XSLT: The Source Tree

The nodes of the source tree correspond to the elements, attributes, text, comments and instructions in the XML document. The nodes are arranged according to the hierarchy in the original XML document (Clark, 1999).

The root node is at the top of the tree – this is the entire XML document.
Children of the root node include the document element and any comments or processing instructions that are not included in the document element (Clark, 1999).

Elements that contain other elements or text are represented as parent nodes with the sub-elements and text represented as child nodes. Attributes of elements are represented as separate nodes as well – but while an element is represented as an attribute node's parent, the attribute node is not the child node of the element (Clark, 1999).

2.3.11 Transforming XML with XSLT: The Templates

Each XSLT stylesheet is made up of a number of templates. These templates are a list of rules for how the source document should be transformed. Each template does two things.

- It specifies which node the template refers to.
- It specifies what should be done with that node or its children.

The XSLT processor starts by building the node tree of the XML document it is processing. Then it starts at the root element and looks for a template to match it. If it finds such a

template, it processes the content of the node the other elements in the XML document - according to the instruction in that template to generate output (Tennison, 2004).

In other words, the XSLT processor generates a node-tree based on the XML document. It works through these nodes, starting at the root node. At each node it looks in the XSLT document for a template that matches that node.

It gets this information from the <xsl: template match = ""> instruction in each template. This instruction specifies what node in the tree that template refers to. The value of the match attribute is an XPath node-set expression that points to a particular node in the XML tree (Tennison, 2004). For example, <xsl: template match – "/"> refers to the root node, and <xsl: template match = heading"> refers to elements named "heading".

Once it has been established which template matches the node in question, that node and its descendents are replaced by the contents of the template. If the template included XSLT instructions, these can change or control the relevant XML fragment. Such instructions might control iteration, flow, or invoke a template matching internal or other nodesets.

If the processor cannot find a template that matches a node that it has been told to process - it uses a built in template to process that node. This means that the processor will take all the children of the current node (the node targeted by the template's match instruction) and will apply the instruction in the template to all of them, in document order (Tennison, 2004).
For example, an XML document:

```
<chapter>
  <exercise>
    <heading>Exercise heading</heading>
    <content>Exercise Content</content>
  </exercise>
  <explanation>
    <heading>Explanation heading</heading>
    <content>Explanation Content</content>
  </explanation>
  <explanation>
    <heading>Second Explanation heading</heading>
    <content>Second Explanations Content</content>
  </explanation>
    <instruction>instructions 1</instruction>
    <instruction>instructions 2</instruction>
    <instruction>instructions 3</instruction>

</chapter>
```

Might have the following XSLT stylesheet:

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <!-- The following template matches the root node -->

  <xsl: template match="/">
    <!-html elements are inserted -->

    <html>
      <head>
        <title>A document</title>
      </head>
      <body>
        <h1>Chapter One</h1>

  <!-The apply-templates instruction invokes the built in template so that all
child       nodes in the document will be inserted between the "body" tags -->

        <xsl: apply-templates/>
      </body>
  </html>
  </xsl: template>

  <!-The template below uses the "match" instruction to target all "heading"
elements which are children of "exercise" elements. The <h1> tags are inserted
around    the content of such elements to alter their appearance in the browser --
>

  <xsl: template match="exercise/heading">
    <h1><xsl: apply-templates select="text ()"/></h1>

  </xsl: template>

  <!-The template below does virtually the same as the previous template except
that    it targets "heading" elements that are children of "explanation" elements,
and a    different html tag is inserted  -->

  <xsl: template match="explanation/heading">
    <h2><xsl: apply-templates select="text ()"/></h2>

  </xsl: template>

  <xsl: template match="instruction">
    <ul>
      <li><xsl:apply-templates select="text()"/></li>
    </ul>
  </xsl:template>

  </xsl:stylesheet>
```

As can be seen in Figure 3, if the XML document shown above were transformed by this XSLT stylesheet, the resulting document would be an HTML document - because of the HTML elements inserted by the first template. The various headings would be displayed suitably formatted to different sizes, and the content of all the "instruction" elements would be displayed as a bulleted list.

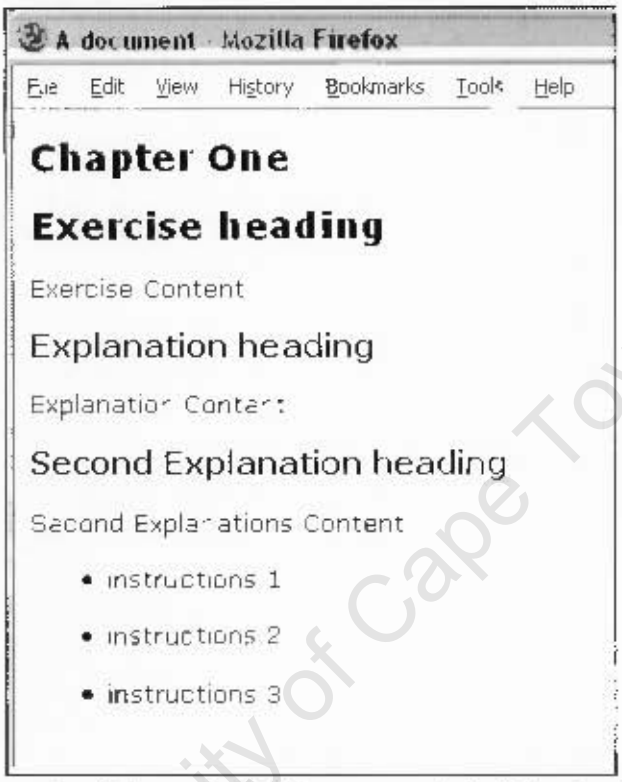

Figure 3: A browser view of the sample XML document styled with the given XSLT stylesheet.

2.3.13 The characteristics of XSLT as a language

XSLT is a *declarative* language as opposed to a *procedural* one. The required transformation is described, and then carried out by the processor. In a procedural language the transformation would be achieved by a specific sequence of procedural instructions. In XSLT the template rules are not arranged in any particular order and don't have to match the order of the input or the output (Kay, 2003).

This is possible because it was designed to be "side-effect free". Side effects occur when it is possible for a function to change its own environment. For example - a function can change the value of global variables that other functions can read. Other examples of side effects could be writing messages to a log file, or prompting the user. It can be crucial, when using such a procedural language, to call the functions the right number of times in the correct order, because each function call creates changes, and will give different results depending on

when and how many times they are called. On the other hand functions that have no side effects can be called any number of times in any order and give the same result (Kay, 2003).

XSLT does not allow you to update the value of a variable. This means that certain solutions which are common in conventional programming cannot be used in XSLT. For example, in a language such as Java or C++ a developer can use iterative loops in which the value of a variable is constantly updated, and used to keep track of the progress of the loop. This is not possible in XSLT. Other solutions, such as using recursive functions, have to be used (Kay, 2003).

2.3.13 Asynchronous JavaScript and XML (AJAX)

In early 2005, the practice of using a combination of technologies to create interactive web application was given the name Asynchronous JavaScript and XML - or Ajax (Garret, 2005). Technologies such as XML, XSLT and JavaScript are used together to update small sections of information on a web page instead of re-loading the entire page. The result is faster interactivity. This was made possible by the use of the XMLHttpRequest object. The AJAX method might be described as:

- Presenting the information using XHTML or CSS.
- Using a client side scripting language such as JavaScript to access the Document Object Model (DOM) and so doing dynamically display and update the presented information.
- Using XML and XSLT to manage the interchange and manipulation of data.
- Taking advantage of the asynchronous data retrieval capabilities of the XMLHttpRequest object. (Garret, 2005) (Mozilla, 2007)

In essence, the initial web page is a JavaScript "AJAX engine" which renders the user interface and communicates with the server. The XMLHttpRequest object allows this communication to happen asynchronously - it can run independently of communication with the server. Instead of all requests being sent to the server via HTTP, requests can be handled by the AJAX engine. In some cases such as simple data validation, requests can be handled by the AJAX engine on its own. In some cases the AJAX engine will get information from the server. In either case, the asynchronous communication of the AJAX engine means that the interaction is not halted at any stage to wait for a response from the server (Garret, 2005). Although this project uses the technologies of JavaScript, XML and XSLT together, it does not use the AJAX techniques of loading data asynchronously as it is not designed to be viewed online, and there is no need to deal with slow download times over the Internet.

2.4 CAI Standards and XML

Since the early 1960s there were many attempts at using CAI by different institutions and organisations. Each created their own content and their own applications to deliver this content. Such applications are called Learning Management Systems (LMS). This soon created problems as the content had to be custom-written for each system (Ostyn, 2005). An example of early LMSs is the PLATO system, the first computer based system designed for general educational use which was introduced in 1960 (Van Meer, 2003). Another is the Computer Assisted Learning Centre (CALC), an offline adult education initiative which started in 1982 (Morabito 2003).

There was much duplication as content on the same subject had to be re-written to suit different systems. If an LMS was upgraded or a new system put in place the content had to be customised to fit the changes. Content could not be re-used or easily moved from one system to another. Producing content was expensive, it was difficult to share content between different LMSs, and upgrading these systems was inefficient, expensive and prone to error. It was necessary to standardise the creation and distribution of content (Ostyn, 2005).

2.4.1 Description of CAI Standards: IMS Metadata

In the late 1990s a number of groups were drawing up specifications for content creation. The EDUCOM consortium of US Institutions of higher education initiated the IMS Project to develop standards for online learning. Other groups in the US National Institute for Standards and Technology (NIST) and the Institute of Electrical and Electronics Engineers (IEEE P.1484) study group were working on similar projects. The NIST projects became part of the IMS Project, and collaborated with the European based ARIADNE Project (IMS, 2004).

In 1998 the IMS and ARIADNE submitted a joint proposal to the IEEE Learning Technology Study Group. This was the beginning of a draft standard for Learning Object Metadata (LOM). Metadata is data about data – in this case tags inserted in the original document that contain information about the content of the document.

LOM is a list of tags that describe learning data in appropriate ways. The IEEE LOM states how metadata documents should be organised. It includes a list of element names, their definitions, data types and field lengths. It also includes guidelines for applications to be considered IEEE conformant (Ostyn, 2005).

The conceptual data scheme of the LOM was finally approved by the IEEE Standards Association in 2002.

2.4.2 Rendering of CAI standards: SCORM

Standards and specifications are open to interpretation. They do not in themselves ensure the creation of durable, portable and reusable content. The Advanced Distributed Learning Initiative (ADL) was launched to deal with possible ambiguity in interpretation. They began by creating a practical profile: the Sharable Content Object Reference Model (SCORM). This was an interpretation of the existing standards and specifications for content such as those produced by the IEEE – the IMS metadata standard and other IEEE standards (Ostyn, 2005).

SCORM focuses on two areas: how to create Web-based learning material that can be delivered and tracked by LMS, and what an LMS must do to deliver and track SCORM compliant learning material. The latest version of SCORM -SCORM 2004 - is conformant with the IEEE standards and the IMS LOM specifications. An example of an LMS that took advantage of these standards and became SCORM compliant is the Mentor LMS (Ostyn, 2005).

2.4.3 Rendering of CAI standards: WHURLE and AtWeb

One example of XML and XSLT used in relation to CAI is an ongoing project at the University of Nottingham, a prototype called WHURLE: Web-based Hierarchical Universal Reactive Learning Environment (Cristea et al, 2005).

The ultimate aim of this project is to create an "Integrated Learning Environment" which responds to learner needs. WHURLE uses transclusion, a process of building a document out of a collection of other smaller documents. Using a process of "conditional transclusion", WHURLE determines what learning material a user should be exposed to according to their user profile. The profile is determined actively by test questions and passively by criteria such as how long a student takes to read a specific page (Cristea et al, 2005).

Another such project, AtWeb, was developed at the University of Warwick (Joy et al, 2002). Like WHURLE, AtWeb chooses the material that is presented to a student based on that student's individual response patterns. They developed a framework for representing the concepts and logic of AtWeb, represented as an XML Document Type Definitions (DTD) which enables the storage of data for the system.

For both WHURLE and AtWeb, it is interesting to note that while the authors speak of the system responding to the student's "learning requirements", they use this term to refer to the

student's ability in a particular subject and level of experience rather than to their learning style (Joy et al, 2002).

2.4.4 Content of CAI standards: DocBook

An example of a popular standard used in CAI among many other areas is DocBook. DocBook is a markup language defined in Standard Generalised Markup Language and XML.

The purpose of DocBook is primarily technical publishing. Its original purpose was the documentation of hardware and software in computer science. However it has since been used in a wider area - to create many types of articles, books, journals and other types of documentation. The core DocBook standard is the DocBook Document Type Definition (DTD) which defines the elements that can be used and how they must relate to one another (Walsh, 2004).

The current organisation that manages DocBook is OASIS, a non-profit global consortium. It is involved in developing e-business standards in areas such as security and electronic publishing (Amorim et al, 2003) (Walsh, 2004).

DocBook is best suited to projects that deal with large quantities of highly structured content. It is not suited to highly design driven content, but rather to any collection of documents that is regularly maintained and published (Walsh, 2004).

The fact that DocBook was originally designed to create support documentation for computer hardware and software - and so is particularly suited to similar applications such as electronic courseware - makes it an appropriate schema for this project. It is also widely used and is a popular schema, supported by many free software environments (Walsh, 2004).

2.5 Summary

There is a general acceptance that CAI has the potential to support the learning environment and enhancing a student's ability to absorb and understand new material. There are also many studies that indicate that such instruction can be enhanced if it is aimed at matching a student's learning style. One way of doing this is to use some level of learner control, whereby the student can control what, when and how they have access to the material. There is much controversy about the advantages of learner control. There seems to be some consensus that a high level of learner control is not always an advantage. Students who are not inherently confident, who are beginners, are immature or suffer from other disadvantages

such as a lack of language, reading or computer skills will not benefit from a high level of learner control.

The effectiveness of CAI is dependent on a number of factors including the manner in which the computer technology is integrated into the learning situation and the context in which the technology is used. It is also important that students be trained to use material that features learner control, and that such material is used in an appropriate context. Lastly, the nature of the material being studied should be considered when deciding on the type and level of learner control.

The work done on creating standards for content management makes it possible to share and update content with greater ease. Combined with this, the various mark-up languages have revolutionised the sharing of information. XML in particular makes it possible to share and re-use information in ways not possible before. It offers the developer the ability to create custom elements to suit a particular situation and it makes data machine as well as human intelligible.

On the other hand, XML is not particularly useful by itself. Something more is needed to take the data which has been so usefully marked, and present it in appropriate forms. Technologies such as XSLT promise to unleash the full power of XML by allowing the transformation of the data in an XML document into other XML documents, or into other formats for use in Web or print publishing. A possible application for this would be in the field of CAI. The ability to transform one set of information into different formats, to change the order, extract certain parts of a document or to change the manner in which the content is presented – all of this has great potential in a field where there is a need to suit content to individual needs. An example of this would be to take a certain amount of content and re-format it in different ways to suit an individual's learning style. Is the combination of XSLT and XML the correct tool for this situation?

Chapter 3: Method, Analysis and Evaluation

In this section the process by which the dynamic notes were created is described and evaluated. Firstly a number of lecturers and trainers were consulted as to the form which the final learning material should take. The feedback from this process became the basis for all decisions made about the dynamic notes.

A basic lesson was written that became the text of the learning material. This text was divided into categories based on the learning dimensions of the Felder – Silverman model of learning styles. The text was marked up in XML, using the DocBook 5.0b5 schema. This involved matching the categories that grew out of the Felder-Silverman learning style model to the elements and attributes available in the DocBook 5.0b5 schema. Where such elements and attributes were not available, the best substitute had to be chosen, and in some cases the schema had to be altered.

Finally the XSLT stylesheets that transformed the XML into HTML were created. To fulfil the demands of the project JavaScript functions had to be added to the HTML page to make it possible for the student to switch between stylesheets and set up the notes according to their learning style preferences.

The resulting dynamic notes were given to a number of people in a pilot test and their feedback was collected in a short questionnaire (Appendix B).

3.1 Interviews and Survey

The first stage was to establish the principles according to which the notes should be constructed. This required getting input from lecturers to establish how they responded to the concept of individual learning styles. It was also necessary to find out more about a lecturer's experience of using computer based learning material as a teaching aid: under what circumstances it was useful and what the drawbacks were. There was also a need to discuss the implications of learning material that can be presented differently depending on an individual's needs in terms of how these should it be presented, and what should be avoided.

The people involved in the initial discussions of the dynamic notes were all lecturers. They work with under- or post-graduate students in universities or private schools, or teach part - time courses for adults.

The discussion took place as face-to-face interviews, or via e-mail as a survey (Appendix A). Each individual was given a summary of the Felder-Silverman learning style model and series of questions around which a discussion was shaped (Appendix A). The questions gauged their response to the idea of learning style models in general, as well as to the Felder-Silverman's model in particular. Further questions dealt with their experience with computer based learning material, as users of such material to gain knowledge for themselves, and as lecturers, using computer based notes in their teaching. All participants were asked to suggest some models according to which the dynamic notes material should be arranged.

The individuals were:

- A history professor at a university who works with under- and post-graduate students and has a special interest in on-line learning and Internet studies.

- A university lecturer in radio journalism who works with under- and post-graduate students.

- A lecturer who teaches marketing strategy at a private tertiary institution to honours students.

- A lecturer in multimedia and digital skills at undergraduate level – also at a private tertiary institution.

- A programmer who teaches short courses on databases and programming to professionals on a freelance basis.

- A university lecturer in English who teaches Critical Context courses to undergraduate students.

- A lecturer who teaches sound engineering to undergraduate students.

The group differed in age: the programmer, the lecturer in radio-journalism and the sound engineering lecturer being of a younger generation than, for example, the English lecturer.

3.1.2 The Felder-Silverman Learning Style Model

Most of the lecturers reacted positively to the Felder-Silverman learning style model. Several of them felt that some of the learning dimensions applied to their own learning style.

The radio-journalism lecturer took issue with the way the learning dimensions are described – there was some discussion around whether "practical application" and "theory" are political and ideologically loaded terms. She suggested that a student's preference for information about practical application as opposed to theory could be as a result of immaturity of thought and experience rather than an indication of learning style. Focussing on practical application could also be as a result of a student's view of studying purely as preparation for "a job" rather than a process of gathering cognitive as well as practical skills.

She also echoed Felder's own comments (Felder, 2002) on the inadvisability of teaching someone only in the learning style they are initially comfortable with, and states that teaching is the process of moving students from a "sequential" to a "global" learning dimension.

The sound engineering lecturer was concerned that if the notes – or the process of using the notes in a teaching situation – refer directly to the concept of learning styles in general, or the Felder-Silverman model in particular, this might distort the learning process. An individual's self perception or value judgements on their own perceived learning style would possibly influence the way in which they use the notes. He felt that the notes should cater for the needs of different individuals without making them aware that they could be categorised according to their learning style. For example, it should not be necessary for students to learn about learning styles or to do tests to determine their learning style.

The manner in which the student navigates through the notes should not refer explicitly to the individual's learning style, but rather aim to appeal to a larger range of learning styles than is the case with traditional learning material.

3.1.3 Computer Assisted Instruction

The use of CAI by these individuals was largely related to the size and type of the institution they teach at. With the exception of the lecturer in English, those lecturers connected with a university had the most experience with using computers in a teaching situation.

The lecturers from private tertiary institutions were more likely to use a very limited form of CAI: they create PowerPoints for use during lectures or as study aids for students outside of class time. Another use of computers in the classroom was mentioned by the Marketing lecturer, who got students to create conceptual models using various applications. The freelance lecturers were least likely to use CAI both when teaching or gathering information themselves. Clearly the availability of resources plays a big part in the use of CAI in this group.

3.1.4 CAI in own learning

Everyone expressed a personal preference for learning from books and other printed material as opposed to computer based learning material. The English lecturer felt that books give one the opportunity to return to specific areas and re-read them with greater ease than – for example – a website might. The programmer found that printed material was more suited to his way of gathering information – an unstructured learning by doing and experimenting – because of the ease of flipping between sections as opposed to clicking from page to page on a computer.

He emphasized the ease with which one can "navigate" a book, and the fact that the book is not able to impose a sequence of learning in the way that a CAI system might. "With a book I'll have an idea of some way of changing the worked example, then flip forward to see what I can find to try to implement it."

3.1.5 CAI in teaching

The university lecturers felt that CAI systems are likely to work better than traditional teaching material as their students have a strong resistance to reading printed material. They also mentioned the benefits of efficient content management systems both for resources and student work. The radio-journalism lecturer felt that a CAI interface creates a dynamic relationship between different types of content. Lecture notes, readings and assignment guidelines can be interlinked in a way that is difficult with printed material. Added to this is the ability to integrate feedback and interactive areas such as forums, hopefully encouraging a less passive attitude to learning.

All but the radio-journalism lecturer emphasized the need for some form of sequential guidance. The notes should start with an overview or a brief summary in which fundamental concepts are explained before the student can move on to other areas.

3.1.6 Weighting and sequence of content

Everyone involved in these discussions was concerned that the content be presented so that students could easily find and identify essential information. If there is no suggested sequence in which information is viewed, a student might struggle to understand the material. They might gain a false sense of their own understanding if they are allowed to skip areas that are not as appealing to them. There could also be issues of safety, cost and waste of time if a student can jump directly into doing a task before having read at least a basic description of what they are meant to be doing.

3.1.7 Freedom to re-order and re-organise information

The built-in structure should to be balanced against the needs of different individuals to learn in different ways. There should be great flexibility in terms of the order in which one can view the notes, to suit the individual's purpose and their learning style. The programmer suggested that one should start with an overview of what will be achieved during the lesson, and after that the student can choose whether to try out and experiment by doing set tasks or to read further explanations. All students should do all tasks and read all the text, but the order in which they do this would be up to them.

3.1.8 Indicating the importance of content

The sound engineering lecturer suggested that there could be visual clues pointing out which areas of the content are essential to the understanding of the lesson. Text could be highlighted or placed in separate blocks. If it is possible to hide less essential areas the fact that they are hidden should be indicated by a link or icon.

The decision on which areas to highlight as essential could be according to a general scheme unrelated to learning style: decisions are made ahead of time as to which areas can not be missed out and this applies to all students. Alternatively, information could be highlighted according to the student's preferred learning style: a student could choose to be shown all areas that have a practical application, or all information related to tasks.

3.1.9 Re-using content

The sound engineering lecturer was concerned that the emphasis on providing content suited to different learning styles could cancel out one of the great advantages of CAI –reusable content. He felt that the content presented to each student should be the same, and only differ in the manner in which it is presented – for example the order. This would make it possible to write a section on a particular topic, and "plug" it into the lesson wherever it is needed as opposed to re-writing that section to suit a learning style.

3.1.10 Context in which notes are used

There was some concern by the radio-journalist and the sound engineering lecturer that the plan according to which the dynamic notes would be created could limit the subject areas in which it could be used. How could one avoid a situation where the categories in which the content is divided are too restricting, or are inappropriate to the subject being taught? Would the notes be more suitable to practical or technical subjects? Are the notes intended for use during a traditionally taught lesson or for use during self study outside the classroom?

3.1.11 Summary of Interviews and Surveys

The summary of the suggestions that came out of this process as to the basic principles according to which the notes should be planned include:

1. The sequence in which the notes are presented should not be infinitely changeable – some structure should be pre-set by the instructor.
2. Whatever form the notes take, the sequence and presentation should make it easy for the student to identify core concepts.
3. The core information should always be available in all forms of the notes – it should not be possible for students to skip sections. If information is hidden there should be a mechanism to indicate this and to enable the hidden information to be revealed.
4. The notes should be created in such a way as to avoid the duplication of content.
5. The end result of the dynamic notes could be a printed version to avoid the inherent difficulties of navigating, and to circumvent the challenges of creating a usable interface.
6. There should be no direct reference to learning styles in the notes, and no need for students to know what their own learning style is, or to know about learning styles in general.

Some questions that arose during this process include:

- Could the fact that a student can choose to have notes presented to them in a particular learning style narrow their learning experience by not challenging them enough?
- Would the categories needed for such notes be flexible enough to provide for a wide range of subject matter?

3.2 Conceptual Design of Notes

3.2.1 Combining the Felder-Silverman Model and the Interview Outcomes

The next step was to write the text of a lesson that could be turned into dynamic notes, and to prepare it for marking up with XML. The subject matter of a common Digital Skills lecture was chosen: "Colour Modes in Digital Images". The length of the text was restricted to that of a typical 2 or 3 hour lesson. This text was then processed according to the specifications of the Felder-Silverman learning style model as well as the suggestions that were gathered during the initial feedback session.

3.2.2 The Felder-Silverman Model

The first stage in converting the learning material into an XML document was to divide it into sections and then to specify how each section meets the needs of the relevant learning dimensions. The categories that emerged from this process became the elements that were used to mark up the learning material in XML. The source for these categories was the Felder - Silverman learning style model.

The Felder-Silverman learning style model describes the learning needs of individuals in terms of the *type* of the learning material as well as the *sequence* in which it should be presented (Felder, 2002). To produce a version of the notes suitable to a particular learning style it must be possible to mark which section is suited to which learning style as well as where in the transformed document each section should appear. Each element needed an appropriate attribute specifying its position in the overall structure, depending on the needs of a particular learning dimension.

3.2.3 Choosing Elements based on the Learning Dimensions:

The categories used were based on the needs of each of the learning dimensions of the Felder – Silverman learning style model. A summary of the learning needs of the different learning dimensions follows:

The Perceptual Learning Dimension:

Sensing:

- Need examples, repetitive exercises, and real world analogies to assimilate a lesson.

Intuitive

- Need access to the underlying theory, conceptual and abstract aspects of a lesson.

The Input Learning Dimension:

Verbal:

- Learn best by reading or listening to verbal explanations.

Visual:

- Prefer Images, diagrams, screenshots and other images to consolidate understanding.

Processing Learning Dimension:

Active:

- Learn by activities, experiments and tasks, profit from group work.

Reflective:

- Gather information and reflect upon the consequences. Prefer open ended questions

Understanding Learning Dimension:

Sequential:

- Prefer information presented in small logical step with no necessity to understand entire context.

Global:

- Need to work out own solutions, and be presented with the "big picture", the context of the lesson (Felder, 2002).

Some of these needs are not relevant to this project. For example; an active learner thrives in a group work environment. The global learner needs to work out their own solutions. How the learning material is used in the classroom is beyond the scope of this project. Some of these needs are satisfied by the form taken by most text-based learning material; for example, the verbal learner's need for verbal explanations and the sequential learners need for information presented in small, logical steps. Taking these away, we are left with the following types of information that should be present in the notes:

- *Examples* for information on practical application and context.
- *Extra information* on underlying theory and concepts.
- *Images* for pictures, diagrams and screenshots.
- *Activities* for tasks and exercises.
- *Questions.*

3.2.4 Using attributes to specify sequence

A mechanism was needed to specify the order in which the elements were to be presented. Each element was allocated a series of attributes to identify where in the page it should be displayed depending on the learning style chosen.

Each of these attributes can contain a numerical value reflecting where in the document the element should be displayed. For example, an **example** element might have a *sensing* attribute of 1, and an *intuitive* attribute of 3, specifying that this particular section of text should be displayed first if the student is navigating according to the "sensing" learning dimension, and third for the "intuitive" dimension.

The order in which the lesson is assimilated is only relevant to some of the learning dimensions so it was not necessary to create such an attribute for each learning dimension. The relevant dimensions for which the order of sections was vital were:

- *Sensing*
- *Intuitive*

3. 3 Integrating Interview Outcomes

Once the basic parts into which the content could be divided had been chosen, it was necessary to return to the suggestions that emerged during the initial feedback session. A strategy had to be devised for the dynamic notes so that they adhered to the principles established during that part of the process.

3.3.1 Suggestion 1 – Limited learner control of sequence:

The sequence in which the notes are presented should not be infinitely changeable – some structure should be pre-set by the instructor.

To satisfy the condition that a certain amount of guidance is necessary in terms of the sequence in which the notes are presented, some sections were marked as "always first" using the appropriate attributes. During the transformation, these sections could be re-organised to a certain extent, but certain aspects of the sequence in which they appear could not be over-written, for example, the "general introduction" section should always appear first. This made it possible to ensure that students do not accidentally skip sections, and that the flow of the overall lesson makes sense.

3.3.2 Suggestion 2 – Identify core learning material:

Whatever form the notes take, the sequence and presentation should make it easy for the student to identify core concepts.

While dividing the text into sections, it was necessary to identify those parts of the text which are essential to the understanding of the lesson, and which are extra material – suitable for enriching a students understanding, but not vital in themselves.

To re-arrange the various elements using XSLT it was necessary to choose some elements that acted simply to divide the content into sections. For example, Examples, Activities and Questions could be grouped together in meaningful units in a section. Sections could then be arranged and re-arranged while their child elements maintained their relationship to one another. To fulfil these needs, the following element was used:

- **Section**

These *sensing* and *intuitive* attributes of each **section** element were used to indicate the relative importance of that particular section in the overall structure of the notes.

To make it possible to identify the sections and their relative importance in the lesson XSLT stylesheets were used. In this way, sections of text marked as **title** in the XML document were displayed as headings, and numbered accordingly.

3.3.3 Suggestion 3 - Core material should always be available:

The core information should always be available in all forms of the notes – it should not be possible for students to skip sections. If information is hidden there should be a mechanism to indicate this and to enable the hidden information to be revealed.

The initial feedback study suggested that the student should be able to access all content in the lesson no matter what individual learning style they are using to navigate the document. On the other hand it was necessary to distinguish between content which can immediately be assimilated by the individual and content which they may prefer to go back to at a later stage.

To make this possible some of the elements were given attributes which would specify whether that element should be "hidden" or "shown" by default – depending on the learning dimension the student is navigating by. In the final version of the notes, hidden sections are indicated as such by coloured blocks and text links, so that the student can choose to reveal them at a later stage (Figures 4&5).

Examples of such elements are the "extra information on theory and concepts", "extra images" and "explanations of practical application." The attributes chosen for this role were

- *show* with possible values "true" or "false"
- *position* to contain the position of the element in the node tree so that it can be referenced from the XSLT template

The "hint" to alert the student to the fact that something was hidden was created by a combination of XSLT stylesheets and Cascading Stylesheets. The mechanism that allows the student to show or hide that content as they prefer was created by a combination of XSLT stylesheets and JavaScript.

An example of this, illustrated by Figure 4 & 5 is wherever an "extra information" element occurred, the relevant template in the XSLT stylesheet would check, using a conditional statement, if that element should appear or not depending on the value of the associated *show* attribute. Independent of the value of that attribute the XSLT stylesheet inserts an HTML element "blockquote" which an associated Cascading Stylesheet causes to appear as a blue rectangle. This rectangle would be empty if the element's *show* attribute is set to "false", and filled with the contents of the element if it is set to "true". The XSLT stylesheet also inserts a text link in this blockquote which triggers a JavaScript function that changes the value of the *show* attribute. In this way the viewer can toggle the visibility of the contents of that element.

The final effect of this is that wherever content is hidden, a coloured rectangle appears. A link stating that the student can "show or hide" content is also visible.

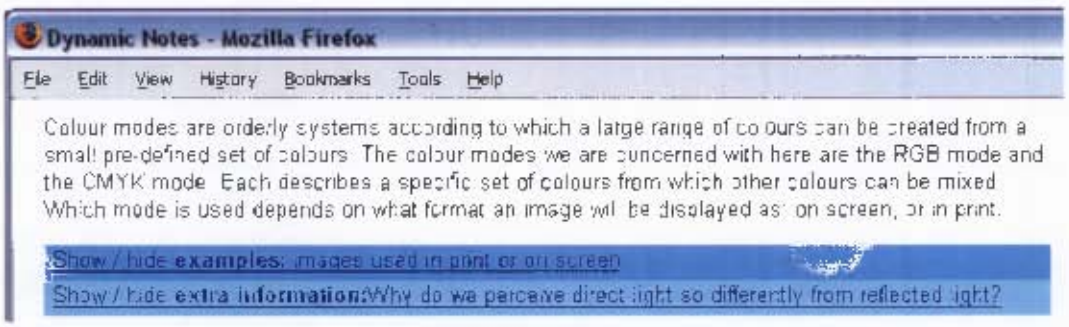

Figure 4: A "show/hide" link before a viewer has clicked on it.

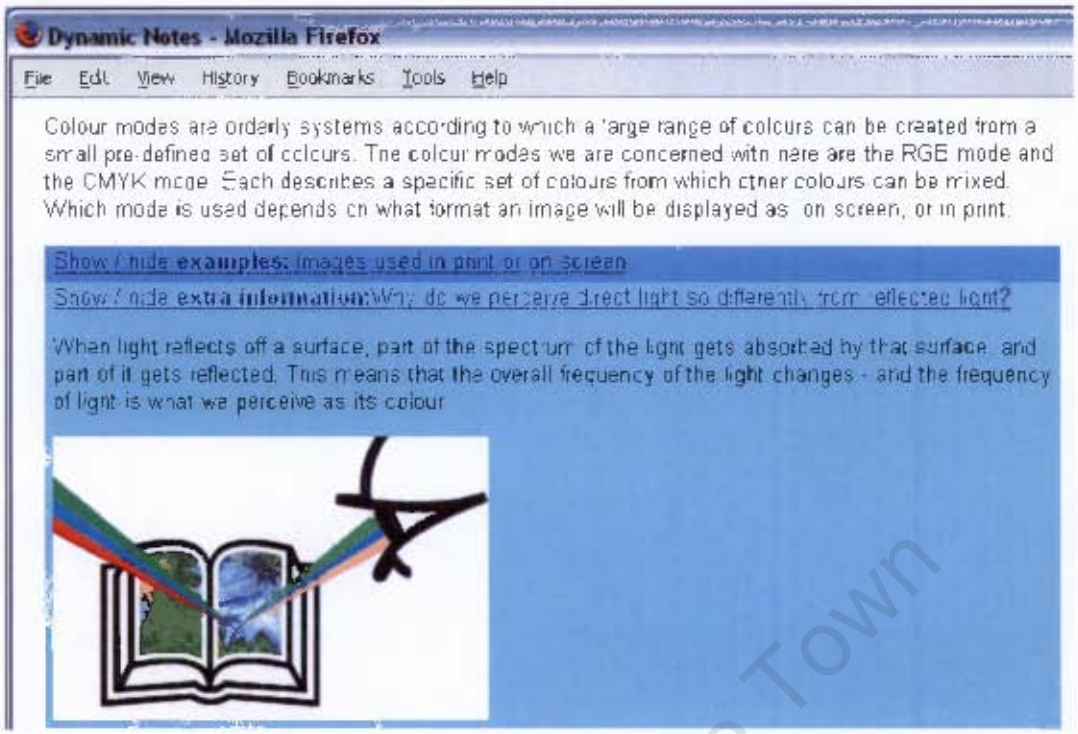

Figure 5: A "show/hide" link after a viewer has clicked on it.

3.3.4 Suggestion 4 - Duplication of content:

The notes should be created in such a way as to avoid the duplication of content.

According to this condition sections of the text were not re-written so that they could be suitable for various learning styles. Instead the technique used to fit the text to the needs of the various learning dimensions was to rely entirely on the order in which the notes' sections are presented and on which areas are shown or hidden by default.

3.3.5 Suggestion 5 - Print vs. On Screen display:

The end result of the dynamic notes could be a printed version to avoid the inherent difficulties of navigating, and to circumvent the challenges of creating a usable interface.

The student using the notes is made aware that the purpose of re-formatting the notes is to prepare a document which will be printed. The purpose of the various buttons and links in the notes are not so much navigation aides, but rather re-formatting aides in the process of preparing a document for printing.

This has implications for the layout of the document as a whole, and for the planning of the user interface. For example, it was not necessary to break the document into "screen-sized"

chunks to avoid the user having to scroll. The issue of navigating from page to page and referring back to previous sections became irrelevant.

These problems are dealt with by means of two preliminary pages which precede the notes: an instruction page (Figure 6) and a "Personalise" form (Figure 7). The instruction page explains the use of the notes, describing the process of re-formatting the notes for print, and the use of the various links and the meaning of the colour coding used.

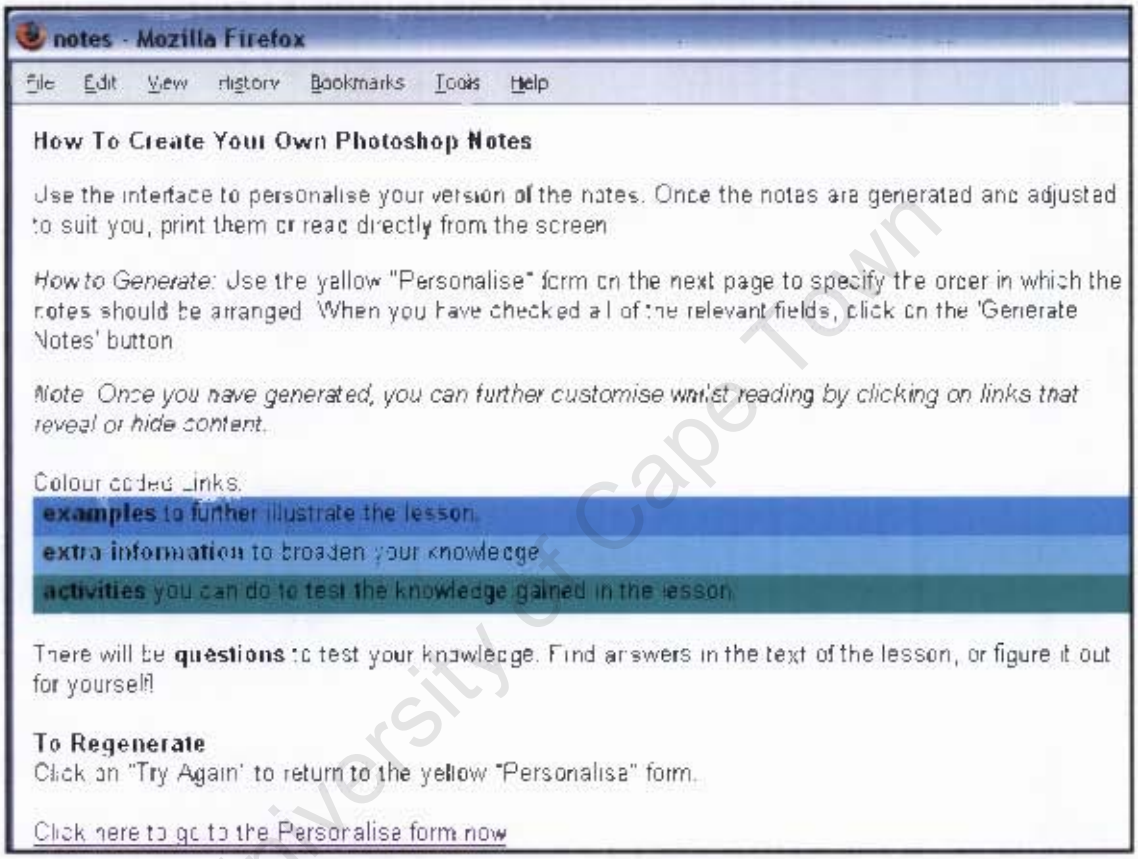

Figure 6: Screenshot of the Instruction Page

3.3.6 Suggestion 6 – Reference to Learning Styles:

There should be no direct reference to learning styles in the notes, and no need for students to know what their own learning style is, or to know about learning styles in general.

This has bearing on the user interface. While there are a number of buttons and links which offer the student a number of options as to how to view the notes, there are no direct references to learning styles, or a need to know about learning styles in order to understand the user interface.

The choice of wording on buttons and links had to be simple and direct, appealing to an individual's need for information presented in a particular form. In other words, a student

who tends towards the sensing learning dimension might want to choose a link which promises to "show me some examples". Their choice would hopefully be motivated by their personal learning style, and not as a result of pre-conceived ideas about learning style.

While avoiding direct reference to learning styles, the structure of the notes was informed by the needs of the individual learning dimensions as described in the Felder – Silverman learning style model.

This was dealt with by means of a page called the "Personalise" form (Figure 7). This is a list of questions that the student has to answer by means of selecting radio buttons. The questions are phrased with no explicit reference to learning styles and a student need have no knowledge of learning styles to choose between them.

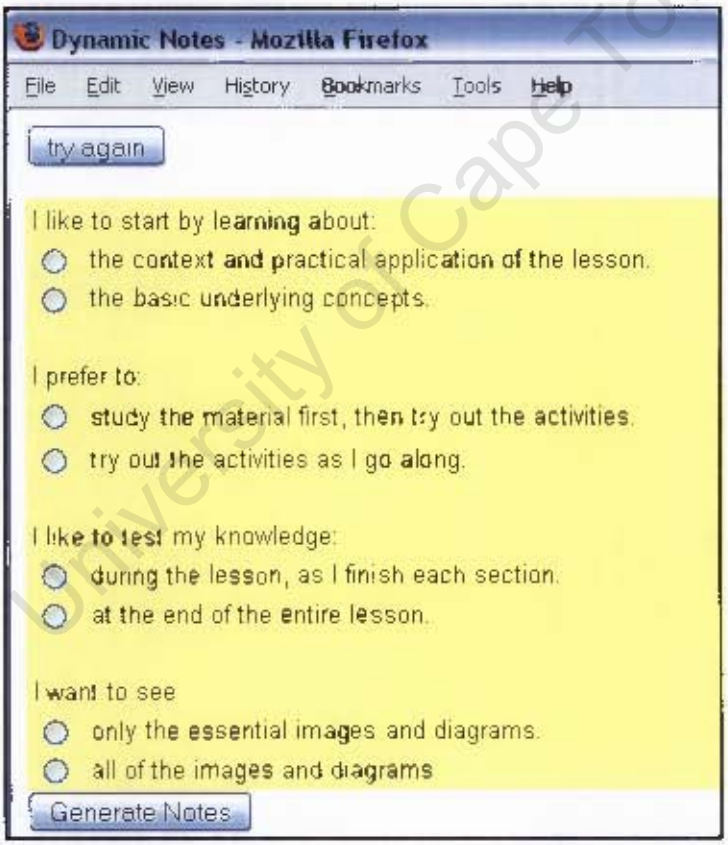

Figure 7: Screenshot of the "Personalise Form"

3.4 DocBook Application Profile

3.4.1 Choosing the categories

The categories that emerged from the Felder-Silverman model that fitted in with the needs of this project were the following:

- *Examples* for information on practical application and context.
- *Extra information* on underlying theory and concepts.
- *Images* for pictures, diagrams and screenshots.
- *Activities* for tasks and exercises.
- *Questions.*

In addition, the following attributes were needed:

- *Sensing and Intuitive* which can take numerical values specifying where in the order the element should appear.
- *show* with possible values "true" or "false" to make it possible to show or hide extra information or images.

During the process of creating the XSLT stylesheets it became apparent that more elements and attributes were needed.

- **section** element

This made it possible to re-organise the text in "chunks" that did not in themselves change order. Having a "section" element made it possible to write simpler XSLT stylesheets. A template could target a parent element directly, but process its children using the built-in template, so it was not necessary to specify the order of each and every element.

- *position* attribute

For those elements which should be "shown" or "hidden" it proved necessary to specify the position of the element in the node tree so that it can be referenced from the XSLT template. This made it possible for a JavaScript function to target a specific element by its position, and so, for example, to hide only a particular **note** element rather than to hide all the **note** elements in the document.

3.4.2 Mapping elements and attributes to DocBook V5.0b5

While most of the elements needed were available in this schema, not all the elements chosen during this process were easy to match to the elements available in DocBook V5.0b5.

DocBook elements used:

Article:

The root element chosen was the DocBook element **article**. An **article** is a general purpose container for articles. It has to contain a **title** element and can contain child element **section** (Walsh, 2004). An example of how this element was used:

```
<article>
   <title>Digital Images: Colour Modes</title>
    <section sensing="1" intuitive="1">
      <!-- All the rest of the elements are nested in the article element -->
    </section>
</article>
```

Section:

The DocBook **section** element is a useful way of dividing the content regardless of its meaning. It has to contain a **title** element, and can contain a number of child elements such as **para, mediaobject, note, qandaset** and **Informalfigure** amongst many others (Walsh, 2004). In the case of this project it was a helpful way of separating different areas of the learning material out from one another so that they could be easily re-ordered using XSLT templates without changing the order of the elements nested within the section element. So the nested paragraphs, images and examples in each section could be presented in a meaningful order, but the sections themselves could be re-ordered.

Each section element was assigned two attributes: *sensing* and *intuitive*, which take numerical values specifying where in the order that section should appear depending which learning style the viewer is navigating by. These attributes are not native to DocBook V5.0b5 and had to be added to the "common attributes" attribute group. An example fragment of XML using showing the use of the section element:

```
<section sensing="3" intuitive="3">

  <title>Primary and Secondary Colours</title>
  <para>All colours can be divided ... by combining  three primaries. </para>

<mediaObject>
  <imageObject>
    <imagedata role="2" show="true" fileref="im/mix.gif"/>
  </imageObject>
</mediaObject>

<note show="false" role="1">
  <title>Why are the primaries not Red, Yellow and Blue?</title>
  <para>Most of us have been taught…</para>
</note>

<qandaset show="true">
  <title/>
    <qandaentry>
      <question>
        <para>Give a concise definition of primary and secondary
colours.</para>
      </question>
    </qandaentry>
  </qandaset>

</section>
```

Para:

Most of the information in the learning material could not easily be categorised as examples tasks or any of the other categories chosen, but still had to be defined as distinct units in each section. The DocBook para element - short for "paragraph"- was the most suitable. An example of how a para element might be used inside a section element:

```
<section sensing="2" intuitive="4">
  <title>What are Colour Modes for?</title>
  <para>When you create an image ... shown on a screen or printed? </para>
  <para>When you look at a printed image ... it via direct light.</para>
</section>
```

Example:

Content that described the real-world application of the lesson or elaborated it further with examples easily fitted into the DocBook **example** element. It requires a **title** element, which is useful in the context of the notes, and can contain **para** elements (Walsh, 2004). As the **example** element was also one of those that could be hidden or revealed depending on the viewer's preference, it was necessary to assign an attribute to the **example** element that would

record whether the element should be shown or hidden. As there was no appropriate attribute in DocBook, the *show* attribute was added to the "common attributes" attribute group in DocBook, and assigned the value of "true" or "false". In addition to this, it was necessary to specify the position of each element in the document so that only a particular element could be affected. The *role* attribute was set to the value of a number which corresponded to the position of that particular element. Some XML to demonstrate the use of the **example** element:

```
<example show="false" role="0">
  <title>Images used in print or on screen</title>
  <para>Print images would be images used in newspapers, ...
presentations.</para>
</example>
```

Mediaobject:

All images were contained in the DocBook **mediaobject** element – an element that is intended to contain external objects such as video, audio or picture information. **Mediaobject** elements must contain a child **imageobject** which must contain a child **imagedata** element. The imagedata element has a *src* attribute to specify the location of an external graphic file (Walsh, 2004).

As many of the images in the transformed document could be hidden or shown depending on the user's preference, it was necessary to assign an attribute to the **imagedata** element that would record whether the element should be shown or hidden. As with the **example** element, the *show* attribute that was added to the "common attributes" attribute group was also used here, and assigned the value of "true" or "false", as can be seen in this example:

```
<mediaobject>
  <imageobject>
    <imagedata show="true" fileref="im/printglue.gif"/>
    </imageobject>
  </mediaobject>
<mediaobject>
```

Task:

The closest fit for the category "exercises and tasks" was the DocBook **task** element. According to DocBook a task element has to contain a child **procedure** element, and nested **step** elements (Walsh, 2004). There are alternative elements which can be nested in a task element but this was the most useful configuration for the purposes of this project. The *show* attribute mentioned in the context of the **mediaobject** element is used slightly

differently in the context of the **task** element, as it is used to check whether or not this element should be displayed in the body of the lesson, or at the end (Walsh, 2004). Here is an example of the **task** element in use:

```
<task show="false">
  <procedure>
  <title>Exploring the "colour recipes" of images in Photoshop</title>
    <step>
     <para>Open "RGB.jpg" image in Photoshop.</para>
    </step>

    <step>
      <para>Open the Info palette an... Magenta (bright pink).</para>
    </step>

    <step>
      <para>Choose the Eyedropper tool... from colour to colour.</para>
    </step>

    <step>
      <para>You can see from this how ... "RGB" colour mode.</para>
    </step>

  </procedure>
</task>
```

Note:

For want of a more appropriate element, sections of text containing extra information on theory or the basic concepts of the lesson were put into the DocBook **note** element. According to the DocBook reference "A **note** is an admonition set off from the main text" (Walsh, 2004). The **note** element was also one of those that could be shown or hidden, and so took the show attribute. The **note** element was a helpful element to use in this context as it allowed for a **title** child element, and it is possible to nest **para** and **mediaobject** elements in a **note** element as can be seen in this example:

```
<note show="false" role="1">
  <title>Why are the primaries not Red, Yellow and Blue?</title>

  <para>Most of us have been taught...</para>

<mediaobject>
  <imageobject>
    <imagedata show="true" fileref="im/printblue.gif"/>
    </imageobject>
  </mediaobject>
<mediaobject>
```

```
</note>
```

QandAset:

Another easy fit of category to DocBook element was the questions to the **qandaset** element. DocBook demands that it contain the child elements **qandaentry** which in turn has child **question** elements to contain the actual questions. The **answer** element is optional (Walsh, 2004). The *show* attribute mentioned in the context of the **mediaobject** element is used in the same way as mentioned in the discussion of the **task** element, as it is used to check whether or not this element should be displayed in the body of the lesson, or at the end. An example of the **qandaset** element in use:

```
<qandaset show="true">
  <title/>
    <qandaentry>

      <question>
        <para>Give a concise definition of primary and secondary
colours.</para>

      </question>

      <question>

        <para>When performing activity x, ...  How can this be?</para>

      </question>

    </qandaentry>

  </qandaset>
```

Informalfigure:

The DocBook **informalfigure** element is intended for graphics with no titles (Walsh, 2004). In this project it was used for the "extra images" category. Like the **mediaobject** element, it was assigned attributes *show* and *role* to keep track of whether the graphic should be hidden or displayed. An example of an **informalfigure** element in use:

```
<informalfigure show="false" role="0">
    <mediaobject>
      <imageobject>
        <imagedata fileref="im/mixlight.gif"/>
      </imageobject>
    </mediaobject>
</informalfigure>
```

3.5 Analysis of using DocBook

The DocBook schema was a useful and efficient tool. It is straightforward to use and understand. It is easy to find documentation and help on how to apply this schema in the form of online documentation and forums. While it provides a convenient structure to keep the XML document well formed and valid, it is also flexible enough to meet the needs of this project.

3.5.1 Number of elements and the level of nesting

Sixteen DocBook elements were used: thirteen compound elements and three leaf elements.

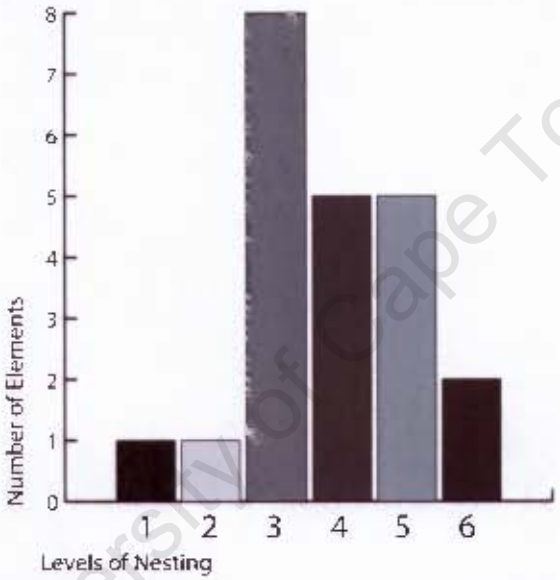

Figure 8: The number of elements at each level of nesting in the DocBook XML document.

As can be seen in Figure 8, there were six levels of nesting, with most of the elements in the third level of nesting or lower, while a fair number (five each) were at the fourth and fifth level. Only two elements were at the sixth level of nesting.

3.5.2 Required Child elements in DocBook

When child-elements are required by the DocBook schema, these are generally appropriate to the context – for example, the **title** element needed in **section** elements which is a convenient way of producing headings for each section. The manner in which elements must be nested is restricted but there are also usually a number of alternative child-elements that can be chosen to suit a particular situation. For example the **task** element could have any of the following elements as a child:

- **blockinfo**
- **example**
- **indexterm**
- **procedure**
- **taskprerequisites**
- **taskrelated**
- **tasksummary**
- **title**
- **titleabbrev**

In the context of this project the most useful option was the **procedure** element, which may contain a large variety of child elements – the appropriate one in this case being **step**.

3.5.3 Nesting in DocBook

As demonstrated in Figure 3 above, most elements were not deeply nested, with most elements being in the third level of nesting. However, in some cases the manner in which elements had to be structured caused a deeper level of nesting than might otherwise have been possible. For example, if you wish to use the **imagedata** element with its useful *src* attribute to specify the location of an external graphic file, you have to nest it in an **imageobject** element which in turn has to be nested in a **mediaobject** element. This made it slightly trickier to perform the necessary transformations using XSLT, as in some cases the XSLT template had to refer directly to the **imagedata** element, and in some to one of the parent elements. In general, though, it was possible to create an XML document with relatively shallow nesting which made it easier to read and understand. This is helpful as much to the writers as to those creating stylesheets to manipulate the document.

3.5.4 Changes made to DocBook

Most of the elements that were needed for this project were available in DocBook. Many of the DocBook elements could be used in a very flexible manner which was helpful: for example the **section** elements and many others were very appropriate, such as the **task** and **example** elements. Some of the elements were an uneasy fit – for example, using the **note** element for the category "extra information on underlying theory and concepts" is a rather arbitrary solution.

The only area in which this schema had to be customised was in providing appropriate attributes. Project specific attributes such as *sensing* and *intuitive* had to be added, as well as more general purpose attributes such as *show*. It would have been possible to use existing attributes for different purposes than they were intended, as for example when using *role* to specify the position of the element – but this would have created a very confusing document semantically. Adding the needed attributes to the schema was straightforward.

3.6 Analyses of stylesheet creation

XSLT is a declarative programming language. A program that is written in a declarative language is a set of rules that specifies what is to be done and not how it is to be done. (Coenen 1999)

3.6.1 Problems experienced with the declarative nature of XSLT

The first barrier in using XSLT is the difficulty in learning the language. For simple transformations from XML into HTML involving the re-organising of data and changing the presentation of the document, using XSLT is simple enough. For the more complex transformations needed in this project the declarative nature of the language was a major stumbling block. Creating different versions of documents without updating variables, being able to use variables in iterative loops or to check the status of the document was a great challenge.

Possible strategies for creating the dynamic notes

This project requires the HTML document loaded in the browser to be continually re-transformed. If XSLT was not a declarative language this might have been done by changing the values of variables in the stylesheet, allowing the viewer to update the values by the choices they make and displaying or hiding information depending on these values. But XSLT *is* a declarative language so the values of variables cannot be updated.

Strategy 1: Using XUL and JavaScript to Edit the source tree

One approach to creating different HTML versions of a single XML document might be to change the source-tree of the XML document. It is possible to edit the source tree of a document using Mozilla's XML based User interface Language (XUL). XUL provides, amongst other things, a number of scriptable DOM objects that can be created using JavaScript and used to edit the Document Object (XULplanet, 2006).

For example, one might change the value of an attribute using the DOM getAttribute () and setAttribute () methods (XULplanet, 2006). If one could change the value of attributes in the original source tree, the XSLT stylesheet could be set up in such a way as to perform different transformations depending on the value of these attributes. This could be reflected in the final transformed HTML document.

The problem here is that the transformation of the source tree has already occurred and the original HTML has been loaded into the browser. If changes were made in the source tree again, in order for the changes to be apparent to the viewer the XSLT transformation process would have to occur once more and the document loaded into the browser again.

Strategy 2: Switching Style Sheets

Another approach would be to create a number of XSLT stylesheets, each of which describes an appropriate transformation for a particular learning dimension. One could then switch the stylesheet that is transforming the XML. By switching to a different stylesheet, a different version of the HTML document could be produced. But once again unless the process of assigning a stylesheet and transforming and loading the document is repeated the transformation would not be visible in the browser and the viewer would be left with the same HTML document.

Chosen Strategy

The technique used in this project is a combination of these two approaches. Firstly, a number of style sheets that offer different versions of the XML data were created. The viewer can trigger JavaScript functions by clicking on links, which have the effect of assigning a different stylesheet to the document which is transformed and re-loaded into the browser so that the transformation is apparent.

It would be time consuming and inefficient to create style sheets for the many possible combinations the viewer might choose. So the viewer can also – by clicking on buttons or links that trigger JavaScript functions – change the values or attributes in the source tree.

These values are checked by conditional statements in the style sheets to produce different versions of the documents based on the values of these attributes. Once again, the transformation and reloading process occurs so that the changes are visible to the viewer.

3.6.2 Problems with switching stylesheets

As there is no simple way to "switch" from one stylesheet associated with a document to another, neither of these two methods are straightforward. The solution used in this project was to use JavaScript to create various XML document objects and change the association of the stylesheet with these document objects.

When switching style sheets, it is necessary to:

1. Create a new empty XML document object.
2. Load the XML document into the new XML document object.
3. Associate the appropriate stylesheet with this new XML object.
4. Remove all parameters and stylesheets from the XLST processor object.
5. Associate the XSLT processor object with the new XML document and its associated stylesheet.
6. Transform the source tree in the new XML object according to the new stylesheet – creating another XML tree in memory
7. Remove part of the node tree of the document loaded in the browser
8. Replace it with the transformed tree structure.

And each time the XML source tree is updated by the user – for example if the value of attributes is changed - it is necessary to repeat steps 4 - 9.

All of these are achieved using JavaScript functions which reside on the HTML page, and which are triggered by the user clicking on buttons.

3.6.3 The DOM and Memory usage

The Application Programming Interface (API) of the XML parser used in this project was the Document Object Model (DOM) API. According to this model the parser builds a source tree – a node-tree based on the XML input, reading the entire XML document into memory in order to create this source tree. In order to be able to process the source tree, the XSLT processor needs the entire XML document to be loaded into memory. An alternative API, the SAX model, is an event based interface which reads only those sections of the document

which are needed. However, there are no in-browser versions of the SAX processor at the time of writing, and so such a processor could not be used in this project.

When a transformation occurs a particular XSLT stylesheet is assigned to a particular XML Document Object. The entire XML document is loaded into memory in the form of a source tree. It is transformed according to the stylesheet into a result tree, and the resulting HTML document is loaded into the browser.

The fact that the XSLT processor needs to load the entire XML document into memory to create the source tree and creates a corresponding result tree of the entire transformed document is acceptable when dealing with small amounts of data. This provides full read-write access to the entire document, which is an advantage. But when dealing with larger documents, this constant loading and re-loading of all the data could result in an excessive use of memory.

When transforming the document by switching stylesheets and changing the source tree, even more data has to be loaded into memory as additional source trees and result trees are created.

3.6.4 Maintaining Modularity with Push and Pull processing

One of the strengths of XSLT is its potential for modularity. If a stylesheet is modular you can choose which parts need to change and to make those changes without affecting the rest of the stylesheet. (Walsh 2001)

One way to achieve modularity is to use the "push" approach to processing XML. With this approach a template rule is written for each node. The children of such a node are not explicitly called, but processed by the built in template. This means that the nodes get processed in document order. (Kay, 2003)

For example, if an XML document was to look as follows:

```
<article>
  <title> An Example </title>
  <para> First </para>
  <para> Second </para>
  <para> Third </para>
</article>
```

Then a typical "push" type XSLT stylesheet would look like this:

```
<xsl:stylesheet xmlns:xsl=http://www.w3.org/1999/XSL/Transform version="1.0">

<xsl: template match = "book">
  <head>
  <title></title>
  </head>

  <body>
   <xsl:apply-templates/>
  </body>
</html>
</xsl:template>

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

  <xsl:template match="title">
    <h1><xsl:apply-templates/></h1>
  </xsl:template>

</xsl:stylesheet>
```

The built in template rules are invoked by the <xsl: apply-templates> instruction so that the processor will send the context node's child nodes to the relevant template rule.

The push approach is better for maintaining modularity for the following reasons.

- If the source document changes, a push type stylesheet is less likely to fail as it does not rely on certain elements being present in a certain structure (Tennison, 2004).
- It is simpler to read and easier to maintain, as it does not rely on the stylesheet writer having an exact knowledge of the structure of the XML document.
- The content will be processed in document order, and the processor can process these in an optimal way. (Milowsky, 2005)

The alternative approach is the "pull" approach. Templates are written for specific elements, targeted using "for-each" and "value-of". An example of a "pull" type stylesheet for the XML example above might be:

```
<xsl: stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">

<xsl:template match="/">
  <html xmlns="http://www.w3.org/1999/xhtml">
    <head>
    <title></title>
    </head>
    <body>
      <h1><xsl:value-of select="article/title"/></h1>
      <xsl:for-each select="article/para">
        <p><xsl:value-of select="."/></p>
      </xsl:for-each>
    </body>
</html>
</xsl:template>

</xsl:stylesheet>
```

It does not use the built in template rules, but rather uses the <xsl: value-of> and <xsl: for-each> type instructions to access particular elements and explicitly state where they should be in the transformed document.

A pull approach is appropriate when the writer knows exactly which elements are present in the source document. It is particularly useful when the elements need to be presented in a different order than they are in the source document (Tennison, 2004). In simpler versions of such stylesheets, it may be easier to read and maintain than the push type (Williams, 2002). The approach used in this project is the mostly the "pull" approach. This is because the order of the elements in the transformed document has to be very different from that in the source document.

Using the pull approach means that many of the templates have to use named select expressions to target specific elements rather than relying on the built in templates. Every select expression causes a document traversal. This means that using select expressions can become expensive (Milowski, 2004).

3.6.5 Choosing where elements appear

One of the challenges of this project was to create a version of the notes in which certain information could be displayed in one section, and another version in which that same information could be placed elsewhere. For example, if the viewer has chosen to navigate as an "active" learner all task elements are integrated in the text of the lesson (Figure 10). If the

viewer is navigating according to some other learning dimension, the **task** elements are grouped together at the end of the lesson (Figure 9).

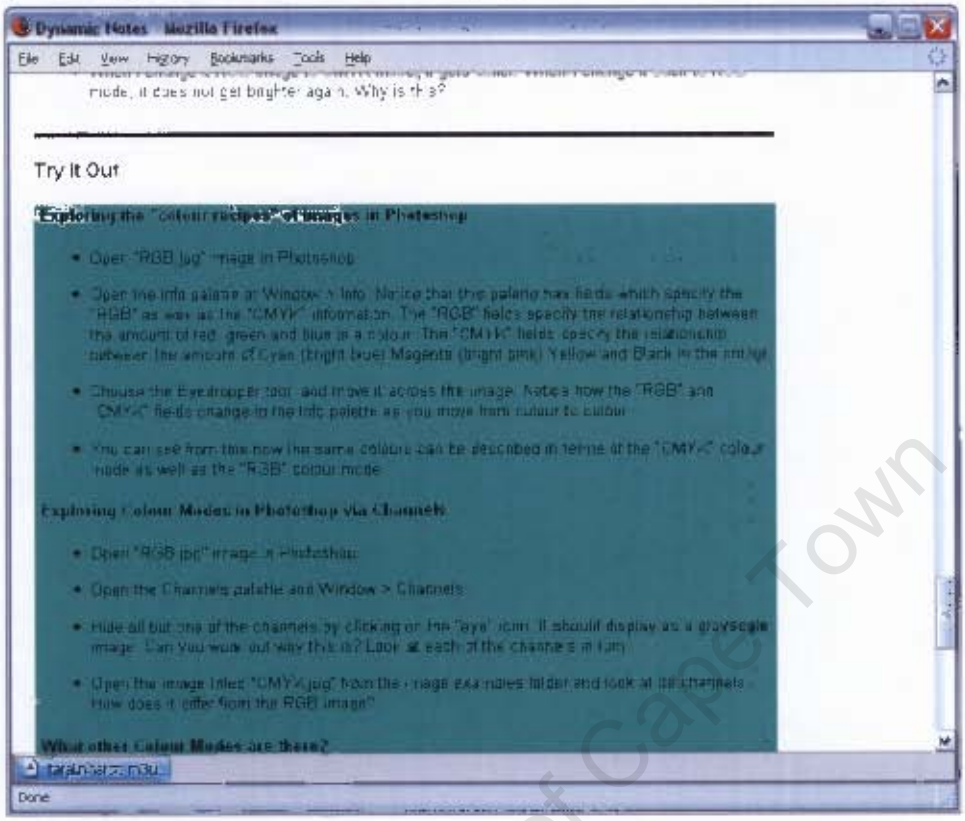

Figure 9: Screenshot of a page in which the viewer has chosen to do all activities at the end of the lesson

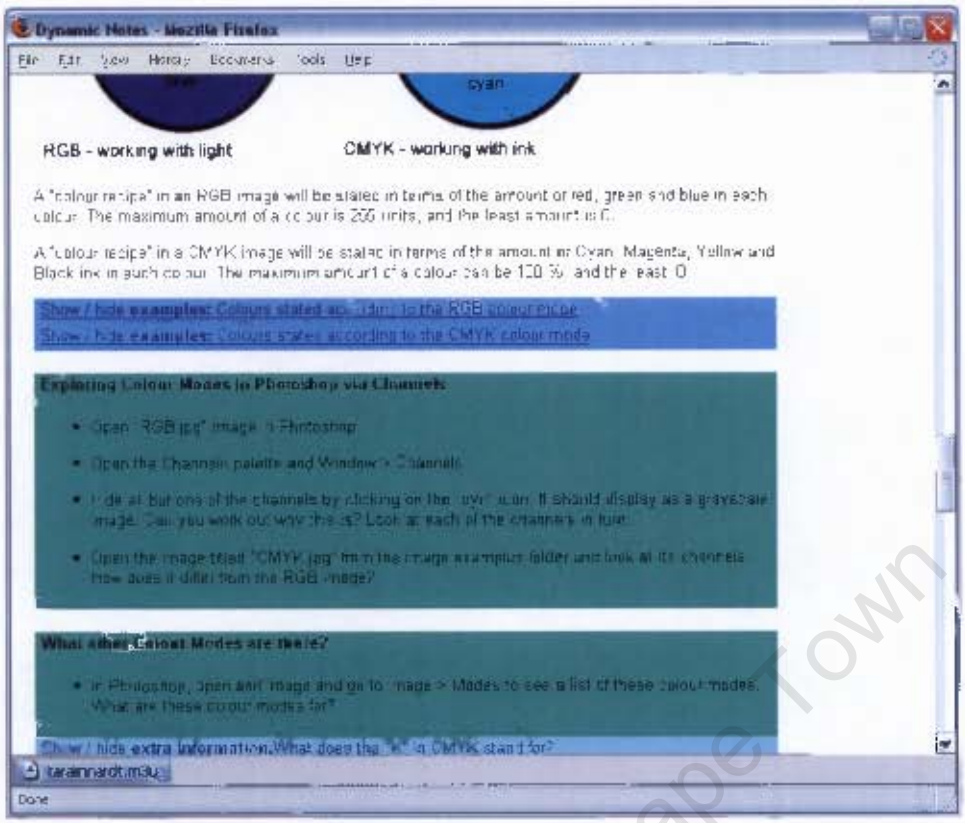

Figure 10: Screenshot of page in which the viewer has chosen to do tasks during the lesson

This was made possible by giving the relevant elements attributes the value of which could be changed via JavaScript by the viewer. In the above example, if the viewer wished to navigate as an active learner, pushing the "active" button would set the *show* attribute of all **task** elements to "true". A conditional statement in the XSLT stylesheet would check whether or not to display the element in a particular place. This would have been the straightforward solution if all that was needed was to show or hide that element. But what was in fact needed was to hide the element in one part of the document, and show it in another.

This was a difficult problem to solve. Most attempts created collisions where two templates target the same element but contradict one another in terms of how that element should be displayed. Other attempts caused the element to be displayed in both positions simultaneously, both to be hidden, or caused all the elements in the document to be displayed several times.

The solution involved a combination of conditional statements and templates aimed at different levels of the node structure.

For example, to process the following XML

```
<section>
  <example/>
  <note/>
  <task/>
</section>
```

in such a way that the task element sometimes appears as a child of that section element and sometimes at the end of the document, the following XSLT had to be created:

```
<xsl:template match="ns:section">

  <xsl:apply-templates/>

</xsl:template>
```

A "parent" template had to be created, which processed all the child elements of a specific element in document order using the built in template – in this case, the **section** element in which the **task** element is nested.

This was convenient and effieient, but resulted in all the child elements (such as the **task** element) appearing in document order whether or not this was appropriate.

To avoid this, a separate template had to be created for those elements that might need to be "hidden" – for example, the **task** element. A conditional statement in this template controlled whether the element was shown or hidden based on the value of an attribute.

```
<xsl:template match="ns:task">
  <xsl:if test="@show = 'true'">

     <xsl:apply-templates  />

  </xsl:if>
</xsl:template>
```

The attribute of that element could be changed – this was done using JavaScript. This ensured that the element could be shown or hidden within the parent **section** element. But to make the element appear at the end of the document, another separate conditional statement had to be created earlier in the stylesheet, in the template that targets the root node:

```
<xsl:template match="/">
//text to create html header,body and other elements

<xsl:apply-templates/>

  <xsl:if test="//ns:task/@show='false'">
     <xsl:apply-templates select="//ns:procedure"/>
  </xsl:if>

</xsl:template>
```

This is not a very satisfactory solution. It will only work if the conditional statement uses the XPath expression with a // (descendant nodes) operator. The // operator forces the processor to visit every descendant, and so is a very expensive operation (Milowsky, 2004).

3.6.6 Browser Compatibility

The methods used to transform the notes only work in specific browsers, and not at all in others. Loading and transforming stylesheets in Internet Explorer is done using ActiveX objects, while Netscape uses its own native TransforMiiX XSLT processor (Mozilla.org). As it stands, the script used in this project only works in Netscape browsers such as Firefox and Mozilla, and generates an error if viewed in Internet Explorer. It is possible to achieve cross browser functionality for a production system with some additional work - for example, providing a JavaScript "wrapper library" that makes it possible to write cross-browser code , but this solution was not implemented in this project as cross-browser (Addey et al, 2002). This was not implemented, as the cross-browser issues are not central to this project.

During the development of this project, it was tested using the browser Firefox 1.0.2 or earlier. When newer versions of these browsers became available a problem became apparent: when the pages are viewed in later versions of Firefox or Mozilla the reset () and transformToDocument () methods have a slightly different effect than that intended. The result is that when a viewer clicks on a link that triggers these methods, the page in the browser re-loads and re-positions itself at the top of the page. On some older laptops, running the program in older versions of the browser caused the program to stop responding.

This is particularly problematic when using the show/hide links in the notes. In the older versions of the browser the viewer could click on the link and the page would re-load in the same position - which has the effect that the page stays unchanged except for the relevant section becoming visible or being hidden. In the newer versions, the page re-positions itself

at the top so that the viewer has to scroll down again to find the section they were originally looking at.

3.7 Usability of Notes

Once a usable version of the dynamic notes was complete it was distributed to a number of people for testing, along with a questionnaire to capture their feedback (Appendix B). Only twelve individuals were involved in the testing, because of limited time and access to potential test subjects. While this means that the results of this test are statistically insignificant, the user test is not the most critical part of the evaluation. The test was to confirm that the notes were usable, but the analysis of the use of XSLT and DocBook is the key aspect of this project.

While most of the people involved in the testing were lecturers, there was also one high school student and two programmers. Each person received a version of the notes and spent some time interacting with the notes: reading and trying them out.
A problem that occurred during testing was that users had different versions of the browser than that used during development, namely Firefox1.0.2. Seven of the twelve individuals used the latest version of Firefox to test the notes - Firefox 2.0 while five used Firefox 1.0.2. In the newer browser the show/hide links (Figures 4 & 5) do not operate entirely as planned, with the page skipping back to the top every time such a link is clicked instead of staying in place on the page as intended.

The decision was made to continue with testing under these circumstances as the purpose of these show/hide links was to prepare a printed document rather than acting as continuously interactive elements.

3.7.1 Navigating of the notes according to learning style

The first test of the dynamic notes was checking to what extent different users would make use of the different navigation options available to them. The options - which are accessed via the "personalise form" in the notes (Figure 7) - are intended to correspond with the different learning dimensions. The assumption is that if the notes are functioning correctly, users would choose a range of options. If the users tended to use the same combination of options this could suggest a problem with this aspect of the notes as it is very unlikely that all the users naturally incline towards the same learning styles.

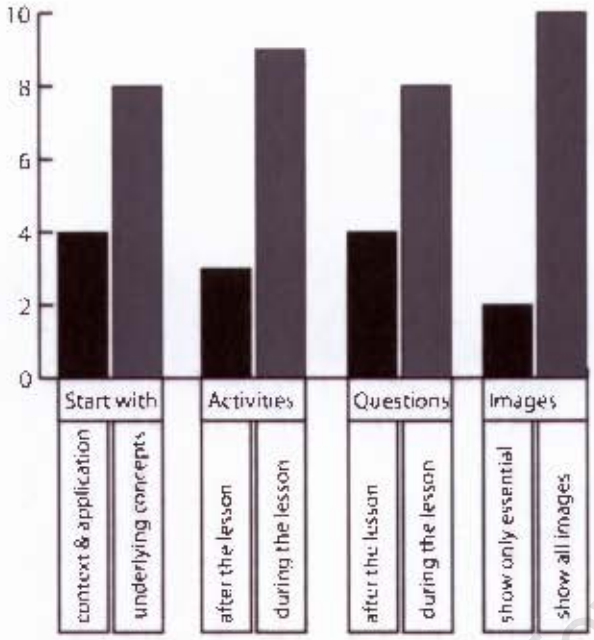

Figure 11: Choices made by users as to how they preferred to navigate the dynamic notes.

Figure 11 indicates the spread of user-choices in the "personalise form" section of the notes. Although there is a definite preference for some options over others, the only area in which there is an overwhelming preference for one over the other is between "show all images" or "show only essential images". This is consistent with the finding that most individuals do tend towards the "visual" end of the "Input" learning dimension (Felder, 2002).

3.7.2 Usability and user experience

The second aspect of the notes that had to be tested was that of usability.

- Did the users feel confident that they understood how the notes should be used: were the instructions (Figure 6) on how to use the notes clear and easy to understand?
- Were there any usability issues with the stage of the notes where users chose to navigate according to their learning style: was the "personalise form" clear and easy to understand?
- Was the text of the lesson itself user-friendly, and were the generated notes clear and easy to understand?
- Were the additional show/hide elements that could be used to transform the notes helpful?
- Did the user feel that the entire experience of being able to generate a personalised version of the dynamic notes added to their learning experience?

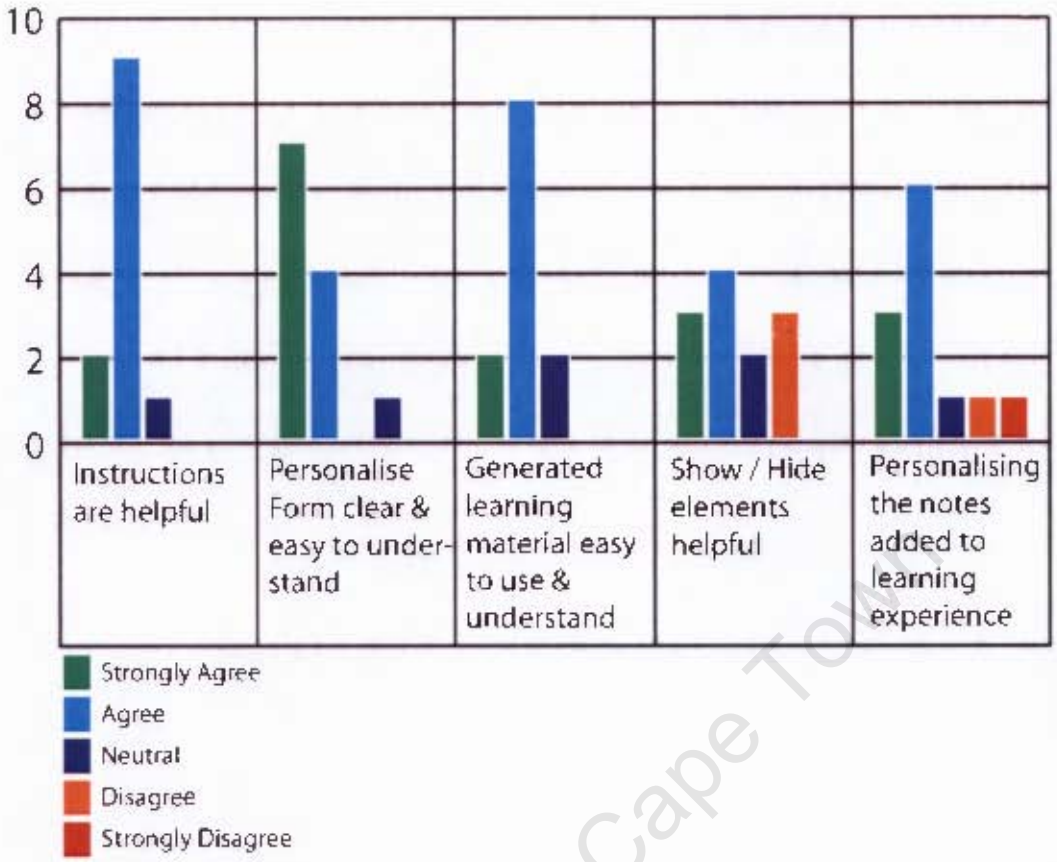

Figure 12: Range of user responses to questions on the usability of the dynamic notes

3.7.3 Feedback on Instructions

There did not seem to be any problems with the initial instructions on using the notes (Figure 6). As shown in Figure 12, while one user chose "neutral" for this option, eleven of the twelve chose either "agree" or "strongly agree" when asked if they found the instructions helpful.

3.7.4 Feedback on Personalise Form

The process of choosing a navigation style (Figure 7) was relatively successful. Seven of the twelve users chose "strongly agree" when asked if they found this section helpful and easy to understand, while four chose "agree" and there was only one "neutral"(Figure 12).

Suggestions made by users included that they would like to have seen examples of a "typical" personalized form. One user felt that there were too many options, and that they might have preferred some form of guidance at this stage - such as an "expert" or a "basic" template to start off with. Another suggestion was that one should be able to save a custom setting of the choices made in a personal profile.

3.7.5 Feedback on the learning material itself

The learning material was well received in this test. There were two potential problems here:

- The content of the learning material could be unclear or confusing enough to make it difficult to determine the cause of a user's problems with the notes.
- The transformation process could render the content difficult to understand, as elements were re-arranged, hidden or shown.

When asked if the generated learning material was easy to use and understand, eight of the twelve chose "agree", with two choosing "strongly agree" and two "neutral" (Figure 12).

Another issue that arose was the manner in which the notes were displayed, as well as the content of the notes themselves. Some users expressed reservations about the lesson, finding it "too long" or "has too many facts", or expressing the need for more images. A user felt that he would have liked to control the size of the "chunks" of information being presented, as "at present, no matter how you customise, you are still presented with a large amount of information at once". Another user suggested the need for a feature by which one could insert comments.

3.7.6 Feedback on the show/hide elements

The reaction on the show/hide elements was mixed and tended to be negative. This may be because seven of the twelve users used different versions of the browser than that used during the development of the notes. This had the effect that the show/hide elements did not operate as intended. When asked whether they found these elements helpful, three of the users who used newer browsers chose "disagree" in this category and two chose "neutral". Of the remaining users, four chose "agree" and three "strongly agree", including two who were using the newer version of the browser (Figure 12).

3.7.7 Feedback on the dynamic notes as a whole

When asked whether the manner in which they could personalise the notes added to their learning experience, six users chose "agree" and three "strongly agree", even though four of them were using the newer version of the browser. One user chose neutral, one "disagree" and one "strongly disagree" (Figure 12). These were three of the users who used the newer version of the browser.

One user felt that the changes that had been made in the notes was not immediately obvious, and would prefer some stronger form of "feedback" to indicate how the notes had been changed.

3.7.8 Summary of Feedback

In general, users responded well to the fact that the notes could be personalised, mentioning in particular that they enjoyed being able to choose whether activities and questions were positioned during or after the lesson (Figures 4 & 5). For example, one commented that the notes were "extremely useful, and may well be great to enhance learner's experience of software training" while another felt that "I liked the fact that I can make these kinds of choices about how the material I am about to learn can be formatted." and "I think overall it's a very interesting concept that could prove to be a great classroom tool".

Users who tested in the newer versions of the browser all expressed frustration with the fact that they had to scroll back down to view the link they had shown or hidden. One user was so frustrated by this problem that she chose not to use this feature at all.

Users chose a range of learning style options when personalising their notes. There was no great preference for a single navigation style and from the comments in the user's general feedback it can be surmised that the choices available to them were useful and added to their learning experience. The aspects that drew particularly positive mention were the ability to choose whether activities and questions should be displayed during or after the lesson.

It is interesting to note that more users chose to navigate according to the more "sensing" learning dimension of "I like to start by learning about the basic underlying concepts" as opposed to the more "intuitive" dimension of starting with the context and practical application. It is generally supposed that more students tend towards the intuitive learning dimension (Felder et al, 1988). This might indicate that the persons testing the notes were not that accurate in gauging their own needs, and made incorrect assumptions about their own preferences. In fact, one of the users had a particularly interesting comment on choosing according to one's learning dimension:

> "I found that my assumptions about the interface that I would prefer was in fact inaccurate.... When I then went back to try the whole interface again using the alternative options, I discovered that they created an interface that was in fact more conducive to my learning. I think that in selecting options students would probably learn as much about their own approach to the assimilation of technical information (and as part of this interrogate their own assumptions with regards to how they learn) as they would about the subjects of the tutorials."

Chapter 4: Conclusions and Future Work

4.1 Summary of what was learnt and evaluation of shortcomings

The main objective of this project was to evaluate the use of XML, DocBook and XSLT to create computer based learning material that could be re-formatted according to the learning style of an individual.

4.1.1 XML and DocBook

XML - specifically of the DocBook schema, was a useful and valuable technology when used in this context. The DocBook schema had to be adapted to add the necessary attributes, and not all of the elements and attributes chosen were an exact fit for the needs of the project. Nevertheless most of the elements and attributes available in the schema were remarkably appropriate for this context.

The concepts involved in using XML and schemas are easily comprehensible even to a beginner or non-programmer. Once one has grasped these basics, using DocBook is straightforward. The documentation on this schema is easily accessible on the Internet, easily understandable and it is also easy to find help on any of the many on-line forums.

The fact that DocBook was developed from the beginning as a tool for creating documentation for software and hardware means that it is admirably suited to the needs of this project. Most of the elements needed were available, and the structure of required child elements and the nesting of such elements does not restrict, but rather adds to the usefulness of the schema. Where the schema had to be customised - for example, in this project some attributes had to be added - this was easy and straightforward to do.

4.1.2 XSLT

While it is clear that XSLT is an immensely valuable tool for the transformation and presentation of XML in general, this technology did not fare so well with the demands made by this project. In this case the transformed document can be quite different from the original XML document. Instead of the content of the XML document being presented in document order, information had to be moved around, hidden and revealed. Of particular importance was that some content had to be moved from one area of the document to another. For example, it was essential that the user could chose whether questions were displayed throughout the document, or grouped together at the end. This type of transformation was particularly difficult to achieve.

The fact that the transformed document differed from the source in such unpredictable ways meant that the "pull" approach to XSLT programming was the dominant approach in this project. Although this is an area of some controversy, many authors feel that the "pull" approach may be problematic in that it results in more complex, less maintainable stylesheets that do not make proper use of some of the inherent strengths of XSLT such as processing the data in a document largely in document order using the built in template (Kay, 2003) (DuCharme, 2005) (Ogbuji, 2001).

The dynamic nature of the notes also required some less than satisfactory solutions. Both strategies employed to make the customisation of the notes possible- that of editing the source tree and switching style sheets - required additional JavaScript to be added to the HTML page. This in itself is not necessarily problematic, but the methods used have several flaws:

- **Browser compatibility**:
    - The technique used to re-assign stylesheets only works in Netscape browsers that use the TransforMiiX XSLT processor, and not in Internet Explorer. The solution is a conditional statement which checks what processor is being used, and to create appropriate code for the relevant processor. This was not implemented as the cross-browser issues are not central to this project.
    - The same applies to support for the technique used to edit the source tree using JavaScript to create DOM objects.
    - Some of the functionality of these techniques only works in certain versions of the browser - specifically in Firefox 1.0.2 and Mozilla 1.7.7 - and don't operate as planned in newer versions.
    - Some older laptop computers caused the program to freeze when run in older versions of the browser.

- **Memory usage:** since the XSLT processor relies on holding all data in memory when transformation takes place, each time the document is transformed the entire XML document tree has to be loaded into memory. Since the document is transformed each time the user re-transforms the notes, this could become problematic. The tree structure in memory can be as much as ten times the original data size (Kay, 2003). For the small document involved in this project, this was still acceptable, but would become problematic for documents of arbitrary size.

4.1.3 User experience

The results of the user-tests of the notes were fairly positive. The concept of having notes that can be re-formatted according to one's learning style was enthusiastically received. Most of the users indicated that they felt that the manner in which the notes were presented and could be re-formatted added to their learning experience. Most of the negative feedback and problems that were experienced by users was the result of them viewing the notes in a version of the browser in which the program did not operate as intended, creating some usability problems. Even so, several of the users who experienced these usability problems still felt that their overall experience was positive.

4.2 Directions for future work

- Improve the usability of the notes themselves: for example, offer a number of different interfaces by which users can "customise" their notes - possibly offering less complex versions of the interface.

- Include better feedback mechanisms to communicate how the user's actions change the presentation of the lesson. This might include appropriate text in headings informing the viewer of changes in sequence or focus.

- Investigate alternatives to the options currently presented in the "Personalise Form". For example: the user might want to choose content according to their level of experience.

- Further user testing must be done to determine to what extent the users' choices in the "Personalise Form" section of the notes are an accurate reflection of their personal learning style.

- At present the purpose of the interface is to design a printed document as this means that there is less of an emphasis on usability in a screen based environment. There could be improvement of the manner in which the notes are presented so that they are more usable in their on-screen version. For example, the user could choose to the have the notes broken into screen sized chunks and navigate among pages.

- Investigate the application of AJAX methods of asynchronous data retrieval to overcome problems with loading and re-loading data in memory.

- Include JavaScript conditional statements to create scripts which work in both Internet Explorer and in the Netscape browsers (Mozilla, 2007).

- Update the appropriate XSLT templates to insert named anchors in the document and so prevent the page-scroll error from recurring in certain browser versions.

4.3 Final Remarks:

Many of the problems encountered during this project, such as the difficulty in coming to grips with the declarative nature of XSLT, were a reflection of a particular individual's experience. Others have found that very aspect of the language to be a benefit (Dodds, 2001). It seems likely that many of the challenges could be surmounted by evolving and emergent Web techniques such as AJAX.

Appendix A: Surveys and Questionnaires used during this project

A.1: Survey used during initial discussion of project.

Background information on Masters research project:
I am creating a set of computer based notes which can be re-formatted by the student to suit their learning style. For example: they can change the way the text is presented on the computer by clicking on a link so that the order of the information is changed, or extra information appears.

The learning style model that I am using is the Felder-Silverman model. According to this there are five "learning dimensions" – five aspects to the process of learning: Perception, Input, Organisation, Processing and Understanding. Each learning dimension is presented as a scale with extreme preferences at either end. For example, the "Input" dimension is presented as a scale between "Very Visual" and "Very Verbal".

A person's overall learning style is a combination of their scores on the scales of all of the five learning dimensions. Here is a description of some of the learning dimensions which I am particularly interested in targeting:

Learning Dimension: Perceptual

Sensing: prefers to be presented with real world analogies rather than symbols or theories. They need to know what the practical application of a lesson is before they feel comfortable absorbing information. They learn best by solving practical problems and repetitive exercises.

Intuitive: prefers to start by understanding the underlying theoretical basis of a lesson. They are comfortable with symbols and concepts and abstract ideas. They are likely to arrive at answers by using creative guesswork based on their understanding basic theoretical concepts. They don't like repetitive work.

Learning Dimension: Processing

Active: prefers to learn while doing. They would experiment, discuss, explain and test rather than observe and think. They learn best by doing hands on tasks and observing the consequences.

Reflective: prefer to gather information and then think about the possible consequences of what they have learnt. They will arrive at answers by reflecting on the new information and drawing conclusions.

Learning Dimension: Input

Visual needs to see diagrams, graphs, pictures, flow charts or timelines to help them understand and remember information.

Verbal prefers to get information verbally – by reading or listening, and by discussing that information. They learn best when they are explaining that information to someone else.

Learning Dimension: Understanding

Sequential: Is most comfortable in being presented with information is small logical steps gradually forming a big picture. They are comfortable learning without understanding what the eventual application of the information might be. They can work with material without

understanding it fully.

Global: need to know what they are working towards when learning. They need to see the context, the big picture and the relevance to their subject from the start. They struggle to work with information that they don't understand completely, and benefit from being allowed to work out their own solutions to complex problems.

APPENDIX A

To Discuss:

- To what extent have you used computer based learning material?

- If you have not – why not?

- Do you prefer working with a book or printed material rather than computer based material? If so, why?

- Imaging that you need to extract information on how to perform a task from a book or computer based text. Suggest ways in which that text should be presented to make it easier for you to get at the information you want.

- Do you feel comfortable with the above description of learning styles?

- If you feel that they are an accurate representation of the way people like to learn: How do you think a lesson / text could be organised to appeal to a person with a strong preference in any of the learning dimension?

- Below is some feedback from other people. Do you have anything to add to this?

APPENDIX A

A.2: Questionnaire used during user testing of notes

1. Please indicate which configuration you found the most useful in each case:

I like to start by learning about:
☐ the context and practical application of the lesson
☐ the basic underlying concepts.

I prefer to:
☐ study the material first, then try out the activities.
☐ try out the activities as I go along

I like to test my knowledge:
☐ during the lesson, as I finish each section.
☐ at the end of the entire lesson.

I want to see:
☐ only the essential images and diagrams.
☐ all of the images and diagrams.

Please indicate you reactions to the following statements:

	Strongly Agree	Agree	Neutral	Disagree	Strongly Disagree
You found the instructions to the notes to be helpful:					
You found the form used to personalise the notes clear and easy to understand:					
The generated learning material was easy to use and understand.					
The interactive elements in the generated notes (show/hide links) were helpful:					
The manner in which you could personalise the notes added to your learning experience:					

Do you have any general comments to make on the experience of using these notes?

Appendix B: The XML document - "notes.xml"

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Masha du Toit (UCT) -->

<article xmlns="http://docbook.org/ns/docbook" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://docbook.org/ns/docbook
schema/docbook.xsd">

    <title>Digital Images: Colour Modes</title>

    <section sensing="1" intuitive="1">
        <title>Introduction</title>
        <para>This lesson is about preparing a digital image for reproduction, in particular, looking at the
accurate reproduction of colour. This lesson looks at the concepts of "colour modes" in various software
programs. You will learn about the various techniques employed to ensure accurate colour
reproduction.</para>

        <para>Any image you create on a computer will ultimately either be printed or displayed on a screen.
To ensure that the picture is reproduced accurately in either of these mediums, you need to understand the
way that colour behaves as reflected light - in the context of ink printed on paper and as direct light - for
example when you view an image on a computer monitor.</para>

        <para>Digital image programs such as Adobe Photoshop have a variety of "Colour Modes" which
make it possible for you to prepare and image for a particular output method. Knowing what these are and
when to use which colour mode will help you achieve accurate and predictable colour reproduction of images.
</para>

    </section>

    <section sensing="2" intuitive="4">
        <title>What are Colour Modes for?</title>

        <para>When you create an image on a computer it is important to keep in mind what medium the
image  will be output to. It might be printed on paper or displayed on a screen.But why should you care
whether an image will be shown on a screen or printed? </para>

        <para>When you look at a printed image you perceive it via beams of light that have reflected off the
page.  When you look at a screen - such as a TV screen, a computer monitor or a cell phone
display, the light shines directly from the screen to your eye. Our eyes perceive colours very
differently when we see it as via reflected light, than when we see it via direct light.</para>

        <mediaobject>
            <imageobject>
                <imagedata show="true" fileref="im/direct.gif" role="1"/>
            </imageobject>
        </mediaobject>

        <mediaobject>
            <imageobject>
                <imagedata show="true" role="1" fileref="im/reflect.gif"/>
            </imageobject>
        </mediaobject>

        <para>To mix a specific colour out of ink requires a certain "recipe" of other ink colours. The same
goes for mixing colours using beams of light.  The important point to understand is that to gain the same
colour, you would need a different "recipe" when using beams of light, than when using ink.</para>

        <para>When you create a digital image the computer keeps a record of each colour in the image.
Each  colour is stated in terms of a recipe -  what colours should be combined to get that colour.  Such a
recipe could be stated in terms of the inks needed, or in terms of the coloured light  needed.  This is what is
meant by the "colour mode" of an image: in what terms the "colour recipes" of that image is stated.</para>

        <para>Colour modes are orderly systems according to which a large range of colours can be created
from a small pre-defined set of colours. The colour modes we are concerned with here are the RGB mode and
the CMYK mode. Each describes a specific set of colours from which other colours can be mixed. Which
mode is used depends on what format an image will be displayed as: on screen, or in print.</para>
```

APPENDIX B

```
<example show="false" role="0">
    <title>Images used in print or on screen</title>
    <para>Print images would be images used in newspapers, books or brochures. On-screen images are those which will be displayed on web-sites, TV screens or slide presentations.</para>
</example>

<task show="false">
    <procedure>
        <title>Exploring the "colour recipes" of images in Photoshop</title>

        <step>
            <para>Open "RGB.jpg" image in Photoshop.</para>
        </step>

        <step>
            <para>Open the Info palette at Window > Info. Notice that this palette has fields which specify the "RGB" as well as the "CMYK" information. The "RGB" fields specify the relationship between the amount of red, green and blue in a colour. The "CMYK" fields specify the relationship between the amount of Cyan (bright blue) Magenta (bright pink) Yellow and Black in the colour.</para>
        </step>

        <step>
            <para>Choose the Eyedropper tool, and move it across the image. Notice how the "RGB" and "CMYK" fields change in the Info palette as you move from colour to colour.</para>
        </step>

        <step>
            <para>You can see from this how the same colours can be described in terms of the "CMYK" colour mode as well as the "RGB" colour mode.</para>
        </step>
    </procedure>
</task>

<note show="false" role="0">
    <title>Why do we perceive direct light so differently from reflected light?</title>
    <para>When light reflects off a surface, part of the spectrum of the light gets absorbed by that surface, and part of it gets reflected. This means that the overall frequency of the light changes - and the frequency of light is what we perceive as its colour</para>

    <mediaobject>
        <imageobject>
            <imagedata show="true" fileref="im/reflect.gif"/>
        </imageobject>
    </mediaobject>

</note>

<qandaset show="true">
    <title/>
    <qandaentry>
        <question>
            <para>Give a concise definition of what a colour mode is.</para>
        </question>
    </qandaentry>
</qandaset>

<qandaset show="true">
    <title/>
    <qandaentry>
        <question>
            <para>We can perceive colours via reflective light as well as via direct light. List some situations in which each of these occurs.</para>
        </question>
    </qandaentry>
</qandaset>
```

APPENDIX B

```
<qandaset show="true">
        <title/>
        <qandaentry>
            <question>
                <para>When performing activity x, I can see that the Info palette in photoshop can display
the "recipe" of a colour in terms of its CMYK as well a its RGB makeup. But the image I used was in RGB
colour mode. How can this be?</para>
            </question>
        </qandaentry>
    </qandaset>

</section>

<section sensing="3" intuitive="3">
    <title>Primary and Secondary Colours</title>

    <para>All colours can be divided into three categories: Primary, secondary and tertiary. Primary
colours are the basic colours: they cannot be achieved by mixing other colours together. Secondary colours
are created by combining two primaries. Tertiary colours are created by combining three primaries. </para>

    <mediaobject>
        <imageobject>
            <imagedata role="2" show="true" fileref="im/mix.gif"/>
        </imageobject>
    </mediaobject>

    <para>The primary colours of light are red, green and blue. You can create all other colours by mixing
together these three colours.</para>

    <informalfigure show="false" role="0">
        <mediaobject>
            <imageobject>
                <imagedata fileref="im/mixlight.gif"/>
            </imageobject>
        </mediaobject>
    </informalfigure>

    <para>The primary colours of ink are cyan (bright blue) magenta (bright pink) and yellow.</para>

    <informalfigure show="false" role="1">
        <mediaobject>
            <imageobject>
                <imagedata fileref="im/paintmix.gif"/>
            </imageobject>
        </mediaobject>
    </informalfigure>

    <note show="false" role="1">
        <title>Why are the primaries not Red, Yellow and Blue?</title>

        <para>Most of us have been taught that the primary colours are red, green and yellow. These
would be the primary coluours of ink (or paint, crayons or marker pens). You were taught that you can mix
blue and yellow ink to produce green ink. You cannot achieve yellow or blue ink by mixing together any other
coloured inks. This means that yellow and blue inks are primary colours, and green is a secondary colour. But
there is a serious flaw in this description of primary ink colours.</para>

        <para>If you were to take the generally accepted ink - primaries: Fire engine red, navy blue and
sunny yellow, you should be able to use these colours to achieve all other colours by mixing them. That is,
after all, what is meant by the term "primary" colours. If you have ever tried doing this, you would know that
things are not so easy. Mixing together red and blue usually results in a quite browny or greyish purple. Blue
and yellow can give you a olivey green. What is wrong here? </para>

        <para>The fact is, the colours usually described as primary - meaning "the basic colours that can
not be achieved by mixing together other colours" are actually secondary, and even tertiary colours. Fire
engine red contains a good amount of yellow. Navy blue contains a large amount of pink. The true primary
colours are much brighter versions of the traditional ones we were introduced to in school: pink instead of red,
cyan instead of blue, and true yellow instead of orange. This is where we get the CMYK colour mode
from.</para>
```

APPENDIX B

```
            <mediaobject>
                <imageobject>
                    <imagedata show="true" fileref="im/printblue.gif"/>
                </imageobject>
            </mediaobject>

            <mediaobject>
                <imageobject>
                    <imagedata show="true" fileref="im/printred.gif"/>
                </imageobject>
            </mediaobject>

        </note>

        <qandaset show="true">
            <title/>
            <qandaentry>
                <question>
                    <para>Give a concise definition of primary and secondary colours.</para>
                </question>
            </qandaentry>
        </qandaset>

        <qandaset show="true">
            <title/>
            <qandaentry>
                <question>
                    <para>What are the primary colours of light?</para>
                </question>
            </qandaentry>
        </qandaset>

        <qandaset show="true">
            <title/>
            <qandaentry>
                <question>
                    <para>Describe what colour is meant by the term "Cyan". How does it differ from the
colour "blue" ?  Why is blue not considered to be a primary colour in ink?</para>
                </question>
            </qandaentry>
        </qandaset>

    </section>

    <section sensing="4" intuitive="2">
        <title>RGB and CMYK</title>

        <para>There are several different colour modes that have been invented to deal with the difference
between a printed image and an on-screen image. The ones we most often deal with are the "RGB" and
"CMYK" colour modes.</para>

        <para>RGB is the colour mode used for images that will be displayed on a screen. RGB stands for
Red, Green and Blue – the primary colours of light. CMYK is the colour mode used for images that will be
printed with ink. CMYK stands for Cyan, Magenta, Yellow and Black.</para>

        <mediaobject>
            <imageobject>
                <imagedata role="5" show="true" fileref="im/rgb.gif"/>
            </imageobject>
        </mediaobject>

        <mediaobject>
            <imageobject>
                <imagedata role="6" show="true" fileref="im/cmyk.gif"/>
            </imageobject>
        </mediaobject>
```

APPENDIX B

```
<para>A "colour recipe" in an RGB image will be stated in terms of the amount of red, green and blue in each colour. The maximum amount of a colour is 255 units, and the least amount is 0.</para>

<para>A "colour recipe" in a CMYK image will be stated in terms of the amount or Cyan, Magenta, Yellow and Black ink in each colour. The maximum amount of a colour can be 100 %, and the least, 0.</para>

<example show="false" role="1">
    <title>Colours stated according to the RGB colour mode</title>
    <para>Bright red is 255r 0g 0b. Each number and letter refers to the number of units of each colour present, so "255r" means "255 units of red", "0g" means "Zero units of green" and so on. A reddish purple can be achieved by mixing 168 red, 0 green and 133 units of blue </para>

    <mediaobject>
        <imageobject>
            <imagedata show="true" fileref="im/purple.jpg"/>
        </imageobject>
    </mediaobject>

    <mediaobject>
        <imageobject>
            <imagedata show="true" fileref="im/red.jpg"/>
        </imageobject>
    </mediaobject>
</example>

<example show="false" role="2">
    <title>Colours stated according to the CMYK colour mode</title>

    <para>A dark Green can be achieved by mixing 67% Cyan, 0% Magenta, 69% Yellow and 47% Black. A dark purple can be achieved with 48% Cyan, 88% Magenta, 0% Yellow and 12% Black.</para>

    <mediaobject>
        <imageobject>
            <imagedata show="true" fileref="im/darkpurple.jpg"/>
        </imageobject>
    </mediaobject>
    <mediaobject>

        <imageobject>
            <imagedata show="true" fileref="im/darkgreen.jpg"/>
        </imageobject>
    </mediaobject>

</example>

<task show="false">
    <procedure>
        <title>Exploring Colour Modes in Photoshop via Channels</title>

        <step>
            <para>Open "RGB.jpg" image in Photoshop.</para>
        </step>

        <step>
            <para>Open the Channels palette and Window > Channels</para>
        </step>

        <step>
            <para>Hide all but one of the channels by clicking on the "eye" icon. It should display as a grayscale image. Can you work out why this is? Look at each of the channels in turn.</para>
        </step>

        <step>
            <para>Open the image titled "CMYK.jpg" from the image examples folder and look at its channels. How does it differ from the RGB image?</para>
        </step>
    </procedure>
</task>
```

APPENDIX B

```
        <task show="false">

            <procedure>
                <title>What other Colour Modes are there?</title>

                <step>
                    <para>In Photoshop, open and image and go to Image > Modes to see a list of these
colour modes.  What are these colour modes for?</para>
                </step>

            </procedure>

        </task>

        <note show="false" role="2">
            <title>What does the "K" in CMYK stand for?</title>

            <para>In theory, mixing together all three ink primaries should result in black.  In practice, the
result is dark grey or brown. This is because of the physical properties of the actual inks involved: they may
not be true primaries, or may not be dense enough.  To achieve true black, it is neccesary to add carbon black
ink when printing - that is what the "K" refers to. </para>

        </note>

        <qandaset show="true">
            <title/>
            <qandaentry>
                <question>
                    <para>What colour mode would you use when preparing an image for on screen
display?</para>
                </question>
            </qandaentry>

        </qandaset>
        <qandaset show="true">
            <title/>
            <qandaentry>
                <question>
                    <para>In the RGB colour mode - the intensity of the colour is expressed in the scale
between 0 and 225, with 225 being the most intense.  In CMYK, colours are stated in terms of percentages:
100% being the most intense.  Why this difference? </para>
                </question>
            </qandaentry>
        </qandaset>

    </section>

    <section sensing="5" intuitive="5">
        <title>Working with Colour Modes</title>

        <para>By now you should know that the CMYK mode is appropriate for images that are to be printed,
while RGB is appropriate for images to be displayed on a screen.  You might think that this means that when
working on an image that is to be printed, you should immediately convert the colour mode to CMYK and do
all your editing in this mode. But this is not the always the case.</para>

        <para>If you are working in a Vector program such as Freehand or Illustrator, working in CMYK mode
from the beginning is a good idea.  This can prevent you from accidentally specifying an ink colour in terms of
the RGB colour mode, resulting in unexpected results when going to print.</para>
        <para>When working in Photoshop, however, you should always do most of your image editing in
RGB mode, and only convert to CMYK to do colour correction before going to print.  With RGB, you have
access to a greater range of the colour spectrum, which makes it possible for you to do finer colour and tonal
adjustments.  A number of image manipulations are not available in CMYK mode.</para>

        <para>When converting an image from RGB to CMYK, you will usually see a definite colour shift.
Many colours that can be described using the RGB mode, cannot be displayed in CMYK mode.  In general,
CMYK colours are duller.  If you consider that CMYK is a approximation of the colours that can be achieved
using ink, while RGB refers to colours mixed from beams of light, this difference in the intensity of colours will
make sense.</para>
```

APPENDIX B

```
<task show="false">
    <procedure>
        <title>Changing colour mode in Photoshop</title>

        <step>
            <para>Open the image titled "RGB.jpg" from the image examples folder.</para>
        </step>

        <step>
            <para>Have a look at the channels pallete (Window > Channels). How many chanels are
there, and what are they called?</para>
        </step>

        <step>
            <para>Go to Image > Mode and choose CMYK from the list.</para>
        </step>

        <step>
            <para>Do you notice any changes to the image? Look at the channels in the channels
palette. What do you notice? </para>
        </step>

    </procedure>
</task>

<task show="false">
    <procedure>
        <title>Testing the colour shift when changing colour modes</title>

        <step>
            <para>Open the Color Palette under Window > Color</para>
        </step>

        <step>
            <para>Use the context menu button ( the little arrow in the top right hand corner) to select
the "RGB" option.</para>
        </step>

        <step>
            <para>Use the sliders to mix  bright a turquoise as you can by setting the "G" and the "B"
sliders to 225 and the "R"slider to 0.</para>
        </step>

        <step>
            <para>You should notice a small "warning" triangle with an exclamation mark appear on
the lower left of the palette.  If you hover over this warning triangle, you will see the message "Warning: out of
gamut for printing".  This means that it is not possible to mix this colour using ink. </para>
        </step>

        <step>
            <para>To get the closest approximation of this colour possible using ink, click on the little
colour square next to the warning.  You should see a significant change in the colour.  </para>
        </step>

        <step>
            <para>This shift is what happens to all the colours in an image if you switch from RGB to
CMYK - the colours are "Brought into printable gamut".</para>
        </step>

    </procedure>
</task>
```

APPENDIX B

```
        <qandaset show="true">
            <title/>
            <qandaentry>
                <question>
                    <para>I am editing an image in Photoshop, which is going to be printed.  What colour
mode should I use when editing this image?</para>
                </question>
            </qandaentry>
        </qandaset>

        <qandaset show="true">
            <title/>
            <qandaentry>
                <question>
                    <para>Why are CMYK image generally duller than RGB images?</para>
                </question>
            </qandaentry>
        </qandaset>

        <qandaset show="true">
            <title/>
            <qandaentry>
                <question>
                    <para>When I change a RGB image to CMYK mode, it gets duller.  When I change it back
to RGB mode, it does not get brighter again.  Why is this?</para>
                </question>
            </qandaentry>
        </qandaset>

    </section>

</article>
```

Appendix C: The JavaScript document -"loader.htm"

```
<html>
  <head>

    <title>Dynamic Notes</title>

    <script language="JavaScript">


var processor;  /* will hold reference to XSLT processor object */
var dataXML;    /* will hold  reference to XML document object */
var xslfile;       /* variable to hold string reference to xslt file*/


 /*Start() function is called  in opening  </body> tag*/

function Start()
{

  processor = new XSLTProcessor();   /* a new XSLT processor object is declared and associated with
variable "processor"*/

   loadData ();  /* call function to create empty XML document object*/

  xslfile = 'stylesheets/basic.xslt'   /* default stylesheet is assigned */
}


/* the loadData() function is called in the Start() function */

function loadData ()
{

  /* creates a new empty XML document object and associates it with
  * the variable dataXML*/

  dataXML = document.implementation.createDocument("", "", null);

  /*below sets the loading of the xml file to be
  * synchronous - the function waits until the document
  * has finished loading before it returns the document
  * and the processing continues.
  *Otherwise it would be necessary to check if the document
  * has finished loading*/

  dataXML.async = false;

/* uses the "load()" method of DOM interface to load the XML document "notes.xml" into the document object
dataXML */

  dataXML.load("notes.xml");


}


/* the WriteHTML function  is called by the buttons at the radio bottom of the page.  The "afile" holds a
*reference to a particular XSLT stylesheet  */

function WriteHTML ( afile )
{
  xslfile = afile; /* the current stylesheet is passed to the WriteHTML function*/
}
```

APPENDIX C

```
/* the refresh() function is called by
*WriteHTML() */

function refresh()
{

/* a new XML document object is created and associated with variable "dataXSL"*/

  var dataXSL = document.implementation.createDocument("", "", null);

/* set loading to be synchoronous */

 dataXSL.async = false;

/*  associate the current xslt file with  dataXSL object*/

 dataXSL.load(xslfile);

/* reset() method of XSLT processor object removes all parameters and stylesheets from the XML object
"processor"*/

  processor.reset();

/*importStylesheet() method  of XSLT processor  imports the stylesheet associated with dataXSL into
"processor"*/

  processor.importStylesheet(dataXSL);

/*"result" is new document object. dataXML is transformed */

  var result = processor.transformToDocument(dataXML, document);

  /*part of the browser document is removed and replaced with the transformed document*/

document.getElementById ("here").removeChild (document.getElementById ("here").firstChild);

 document.getElementById ("here").appendChild(result.firstChild);
}

/* wipe() is called  by the "try again" button,  to remove all parameters from XSLT processor object and reload
the form */

function wipe()
{

  processor.reset();

    var result = processor.transformToDocument(dataXML, document);
    document.getElementById ("here").removeChild (document.getElementById ("here").firstChild);

 document.getElementById ("here").appendChild(result.firstChild);
}


/* changeAtt() changes an attribute in the XML document object,
* "ell" holds a reference to an  element and "att" to the new value of the attribute.
*Triggered by the radio buttons on the form, which are used
* to set the value of the "show" attribute of various elements to
* determine where in the notes they will appear*/

function changeAtt(ell, att)
{

var el;   /*variable to refer to element*/
var list; /* variable which will hold list of elements*/

el = ell;
```

APPENDIX C

```
/*get all elements in dataXML according to the value of el*/

list = dataXML.documentElement.getElementsByTagName(el);

/*loop through the elements and set all attributes
* named "show" to the new value  of "att"*/

for(var i = 0; i < list.length; i++)
{
list[i].setAttribute("show", att);
}

}

/* showHide is called  by the "show/hide" links in the notes,
* to test what the current value of the "show"
* attribute of an element is, and then to
* set it to the opposite value so that the
* element can be toggled between shown or  hidden.
* The "ell" holds a reference to the element name
* while the "number" holds a number
* that will correspond to the position of a particular element */

function showHide(ell, number)
{

list = dataXML.documentElement.getElementsByTagName(ell);

/*the "ell"  in the getAtt function refers to the name of a element
* the "number" is passed the value of  the "role" attribute of that element and is
*used to specify which element is being changed
*getAtt() returns the value of the element referred to with the name "ell" and the
* position "number"*/

a = getAtt(ell, number);  if (a == 'false'){

list[number].setAttribute("show", "true");
}

else if (a == 'true')
{

list[number].setAttribute("show", "false");
}
refresh();
}

/* getAtt() is used to find the value of an attribute of an element.
* used by the showHide() function*/

function getAtt(ell, number){

var el; // the name of the element
var att= ""; //a placeholder for the value of the attribute

//this creates an array of all the elements with the name stored in the variable "ell"
el = dataXML.documentElement.getElementsByTagName(ell);


 /* this specifies a particular element in the array, determined by the value of
 * the variable "number", which is equal in value to the "role" attribute of that element.
 the attributes.getNamedItem () retrieves the value stored in the attributes of that element*/

att= el[number].attributes.getNamedItem("show").value;

return att; // the value of this attribute is returned

}
```

APPENDIX C

```
/* getCheckedValue is used to check whether the
* user has ticked the radio buttons that
* specify which stylesheets should be used */

function getCheckedValue(radioObj) {

    var radioLength = radioObj.length;

    for(var i = 0; i < radioLength; i++) {
        if(radioObj[i].checked) {
        refresh();
            return ;
        }
    }

    alert("Please make sure you have checked all the options.");

}

</script>

<meta content="MSHTML 6.00.2800.1528" name=GENERATOR>

<link href="stylesheets/basic.css" rel="stylesheet" type="text/css">

  </head>

   <body onload=Start()>

    <form >

     <button  onclick="wipe(); return false">try again</button>

   </form>

<div id=here><form name="chooser">

    <table width="600" bgcolor="#FFFFCC" class="normal">
      <tr >
        <td colspan="2" >I like to start by learning about: </td>
      </tr>

      <tr >
   <td width="24" ><input type="radio" value='sensing' name="style"
onClick="WriteHTML('stylesheets/sensing.xslt')"/></td>
   <td width="564" >the context and practical application of the lesson.</td>
  </tr>

  <tr >
   <td ><input type="radio" value='intuitive' name="style"
onClick="WriteHTML('stylesheets/intuitive.xslt')"/></td>

   <td >the basic underlying concepts.</td>

  </tr>

  <tr >
   <td> </td>
   <td > </td>
  </tr>

  <tr >
   <td colspan="2">I prefer to: </td>
   </tr>
```

APPENDIX C

```
<tr >
  <td ><input type="radio" value='true' name="active" onClick="changeAtt('task', 'false')"/>
    <br/>
  </td>
  <td > study the material first, then try out the  activities.</td>
</tr>

<tr >
  <td ><input type="radio" value='true' name="active" onClick="changeAtt('task', 'true')"/></td>
  <td > try out the activities as I go along.</td>
</tr>

<tr >
  <td> </td>
  <td > </td>
</tr>

<tr >
  <td colspan="2">I like to test my knowledge: </td>
  </tr>

<tr >
  <td ><input type="radio" value='true' name="questions" onClick="changeAtt('qandaset', 'true')"/></td>
  <td >  during the lesson, as I finish each section.    </td>
</tr>

<tr >
  <td ><input type="radio" value='true' name="questions" onClick="changeAtt('qandaset', 'false')"/></td>
  <td > at the end of the entire lesson.    </td>
</tr>

<tr >
  <td> </td>
  <td > </td>
</tr>

<tr >
  <td colspan="2">I want to see </td>
  </tr>

<tr >
  <td ><input type="radio" value='true' name="images" onClick="('informalfigure', 'false')"/></td>
  <td > only the essential images and diagrams.    </td>
</tr>

<tr >
  <td ><input type="radio" value='true' name="images" onClick="changeAtt('informalfigure', 'true')"/></td>
  <td > all of the images and diagrams.</td>
</tr>

</table>

<button  onclick="getCheckedValue(document.forms['chooser'].elements['style']); return false">Generate
Notes</button>

</form></div>

</body>
</html>
```

Appendix D: The XSLT Stylesheets

D.1: "intuitive.xsl"

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:ns="http://docbook.org/ns/docbook">


<xsl:template match="ns:article">
<h2><xsl:apply-templates select="ns:title" /></h2>
    <xsl:apply-templates select="ns:section">
        <xsl:sort select="@intuitive" />
</xsl:apply-templates>
</xsl:template>

<xsl:include href="basic.xslt"/>


</xsl:stylesheet>
```

APPENDIX D

D.2: "sensing.xsl"

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:ns="http://docbook.org/ns/docbook">


<xsl:template match="ns:article">
<h2><xsl:apply-templates select="ns:title" /></h2>
    <xsl:apply-templates select="ns:section">
        <xsl:sort select="@sensing" />
</xsl:apply-templates>
</xsl:template>

<xsl:include href="basic.xslt"/>


</xsl:stylesheet>
```

APPENDIX D

D.3 "basic.xsl"

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

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:ns="http://docbook.org/ns/docbook">

<xsl:template match="/">

    <html>
        <head>
            <title></title>

             <link rel="stylesheet" href="stylesheets/basic.css"/>

        </head>

        <body>
        <table width="600" class="normal">
        <tr>
            <td>
                <xsl:apply-templates/>


                <!-- test to see if user has chosen to view all questions at the end of the lesson.  If they have
the attribute "show" will have the value "false" -->


                    <xsl:if test="//ns:qandaset/@show='false'">
                    <hr/>
                        <h3>Test Your Knowledge</h3>
                        <blockquote class="question">
                        <xsl:apply-templates select="//ns:question"/>
                        </blockquote>
                    </xsl:if>


                <!-- test to see if user has chosen to view all activities and tasks at the end of the lesson.  If
they have the attribute "show" will have the value "false" -->


                    <xsl:if test="//ns:task/@show='false'">
                    <hr/>
                        <h3>Try It Out</h3>
                        <blockquote class="task">
                        <xsl:apply-templates select="//ns:procedure"/>
                        </blockquote>
                    </xsl:if>


            </td>
        </tr>

    </table>

        </body>

    </html>
</xsl:template>
```

APPENDIX D

```
<xsl:template match="ns:section">

<hr/>

<h2>Section: <xsl:value-of select="position()"/></h2>
<xsl:apply-templates/>
</xsl:template>

<xsl:template match="ns:para" >
<p><xsl:apply-templates/></p>
</xsl:template>

<xsl:template match="ns:section/ns:title|ns:procedure/ns:title" >
<h4><xsl:apply-templates /></h4>
</xsl:template>

<!-- test to see if user has chosen to view all tasks during the lesson.  If they have the attribute "show" will
have the value "true" -->

<xsl:template match="ns:task">
<xsl:if test="@show = 'true'">
     <blockquote class="task">
     <xsl:apply-templates  />
     </blockquote>
  </xsl:if>
</xsl:template>



<xsl:template match="ns:step">
<ul>
     <li>
<xsl:apply-templates />
</li>
</ul>
</xsl:template>



<xsl:template match="ns:note">

<blockquote class="note">

<!-- creates a link that calls the JavaScript function showHide().  This will  toggle the visibility of the contents of
this element.  It passes the name of this element, and the value of the "@role" attribute to specify which
instance of the element is to be targeted-->

<a href = "#" onclick="{concat (&quot;showHide('note', '&quot;, @role, &quot;') ;  return false&quot;)} ;  return
false" >Show / hide<b> extra information:</b>

<xsl:apply-templates select="ns:title"/> </a>

<xsl:if test="@show = 'true'">
 <xsl:apply-templates select="ns:para" />
<xsl:apply-templates select="ns:mediaobject"/>
  </xsl:if>

 </blockquote>

</xsl:template>
```

APPENDIX D

```
<xsl:template match="ns:informalfigure">

<blockquote class="note">

<!-- creates a link that calls the JavaScript function showHide(). This will toggle the visibility of the contents of
this element. It passes the name of this element, and the value of the "@role" attribute to specify which
instance of the element is to be targeted-->

<a href = "#" onclick="{concat (&quot;showHide('informalfigure', '&quot;, @role, &quot;') ; return false&quot;)} ;
return false" >Show / hide<b> extra information:</b> Image
 </a>

<xsl:if test="@show = 'true'">
<br/>
 <xsl:apply-templates />
  </xsl:if>

 </blockquote>

</xsl:template>



<xsl:template match="ns:example">

<blockquote class="ex">

<!-- creates a link that calls the JavaScript function showHide(). This will toggle the visibility of the contents of
this element. It passes the name of this element, and the value of the "@role" attribute to specify which
instance of the element is to be targeted-->

<a href = "#" onclick="{concat (&quot;showHide('example', '&quot;, @role, &quot;') ; return false&quot;)} ;
return false" >Show / hide <b>examples: </b>
<xsl:apply-templates select="ns:title"/> </a>
<xsl:if test="@show = 'true'">

 <xsl:apply-templates select="ns:para" />
<xsl:apply-templates select="ns:mediaobject"/>
  </xsl:if>

 </blockquote>

</xsl:template>



<xsl:template match="ns:qandaset">

<!-- test to see if user has chosen to view all questions during the lesson. If they have the attribute "show" will
have the value "true" -->

<xsl:if test="@show = 'true'">
<blockquote class="question">
<b> Question <xsl:number/> </b>

<xsl:apply-templates />

 </blockquote>
 </xsl:if>

</xsl:template>
```

APPENDIX D

```
<xsl:template match="ns:question">
<ul>
    <li>
<xsl:apply-templates />
</li>
</ul>
</xsl:template>



<xsl:template match="ns:imagedata">
<img >
<xsl:attribute name="src">
<xsl:apply-templates select="@fileref"/>
</xsl:attribute>
</img>

</xsl:template>

</xsl:stylesheet>
```

Bibliography

Addey, D & Auld, C & Gudmundsson, O & James, J & Kent, A & Rafter, J & Shiell, A & Spencer, P & Surguy, I (2002) 'Practical XML for the Web', Birmingham, Glasshaus.

Advanced Distributed Learning (2005) 'Product Information' [Online] Available: http://www.adlnet.gov/scorm/certified/index.cfm?event=main.product&certid=149 [accessed 3/9/2006]

Amorim, R & Lama, M & Sanchez, E & Villa, X (2003) 'Advances in Technology-based Education: Towards a Knowledge Based Society', *Proceedings of the II nternational Conference on Multimedia and Information & Communication Technologies in Education,* [Online] Available: www.formatex.org/micte2003/micte2003.htm [accessed 11/4/2006]

Anderson, T (2004) 'Introducing XML', *Tim Anderson's IT Writing* [Online] Available: http://www.itwriting.com/xmlintro.php [accessed 3/9/2006]

Bajraktarevic, N & Hall, W & Fullick, P (2003) ' Incorporating learning styles in hypermedia environment: Empirical evaluation' *Proceedings of the Fourteenth Conference on Hypertext and Hypermedia, Nottingham.* [Online] Available: <http://wwwis.win.tue.nl/ah2003/proceedings/paper4.pdf > [accessed 10/5/2005]

Bray, T Pauli & J Sperberg-McQueen, C & Maler, E & Yergeau,F (2006) ' W3C Recommendation: Extensible Markup Language (XML) 1.0' Fourth Edition [Online] Available: http://www.w3.org/TR/2006/REC-xml-20060816/ [accessed 3/7/2006]

Clark , J (1999) 'W3C Recommendation: XSL Transformations (XSLT) Version 1.0' [Online] Available: http://www.w3.org/TR/xslt [accessed 3/7/2006]

Coffield, F & Moseley, D & Hall, E & Ecclestone, K (2002) ' Should we be using Learning Styles?' [Online] Available from www.lsda.org.uk/files/PDF/1540.pdf [accessed 20/5/2005]

Cristea, A & Stewart, C & Brailsford, T & Cristea, P (2005) 'Evaluation of Interoperability of Adaptive Hypermedia Systems: testing the MOT to WHURLE conversions in a classroom setting.' *3rd Adaptive and Adaptable Educational Hypermedia workshop at the Artificial Intelligence in Education 2005 conference, July 18-22, 2005, Amsterdam, The Netherlands at*

the Adaptive Hypermedia 2004 conference, Eindhoven, The Netherlands [Online] Available: http://portal.acm.org/citation.cfm?id=504259&dl=ACM&coll=&CFID=15151515&CFTOKEN=6184618 [accessed 11/4/06]

Dodds, L (2001) 'Architectural Style', *XML.com* [Online] Available: http://www.xml.com/pub/a/2001/08/15/architecturalstyle.html [accessed 8/13/2006]

DuCharme, B (2005)' Push, Pull Next', *XML.com* [Online] Available: http://www.xml.com/pub/a/2005/07/06/tr.html [accessed 10/8/2006]

Felder, R (2002) ' Matters of Style' [Online] Available: <www.ncsu.edu/felder-public/Papers/LS-Prism.htm> [accessed 10/5/2005]

Felder, R & Silverman, L (1988) 'Learning and Teaching Styles in Engineering Education'[Online] Available www.ncsu.edu/felder-public/Papers/LS-1988.pdf [accessed 10/05/2005]

Garret, J (2005) 'Ajax: A new Approach to Web Applications' [Online] Available: http://www.adaptivepath.com/publications/essays/archives/00385.php [accessed 21/1/07]

Grigoriadou, M & Papanikolaou, K & Kornilakis, H & Magoulas G, (2001) 'INSPIRE: An Intelligent System for Personalized Instruction in a Remote Environment' *Proceedings of the Eight International Conference on User Modeling (UM2001)*, (Sonthofen, Germany, 2001), [Online] Available:<http://www.is.win.tue.nl/ah2001/papers/papanikolaou.pdf> [accessed 6/3/2005]

Howard, R & Carver, C & Lane, W (1996) 'Felder's Learning Styles, Bloom's Taxonomy and the Kolb Learning Cycle: Tying it all together" *Proceedings of the twenty-seventh SIGCSE technical symposium on Computer science education*, [Online] Available <http://portal.acm.org/ft_gateway.cfm?id=236545&type=pdf&coll=portal&dl=ACM&CFID=43092425&CFTOKEN=80051003> [accessed 23/3/2005]

IMS Global Learning Consortium (2004) 'IMS Meta-data Best Practice Guide for IEEE 1484.12.1-2002 Standard for Learning Object Metadata Version 1.3' [Online] Available: http://www.imsglobal.org/metadata/mdv1p3pd/imsmd_bestv1p3pd.html [accessed 11/3/2006]

Joy, M & Muzykanskii,B & Rawles, S & Evans, M (2002) 'An Infrastructure for Web-based Computer Assisted Learning' *ACM Journal on Educational Resources in Computing* [Online] Available: http://delivery.acm.org/10.1145/950000/949261/p4-joy.pdf?key1=949261&key2=2186727511&coll=&dl=ACM&CFID=15151515&CFTOKEN=6184618#search=%22AtWeb%20Warwick%22 [accessed 10/5/2005]

Kay, M (2001) 'XSLT 2nd Edition Programmer's Reference' Birmingham, Wrox Press

Lunts, E. (2002) ' What does the Literature Say About the Effectiveness of Learner Control in Computer-Assisted Instruction?', *Electronic Journal for the Integration of Technology in Education* [Online] Available: < http://ejite.isu.edu/Volume1No2/Lunts.htm > [accessed 6/3/2005]

Milowski , R (2004) 'There are Monsters in My Closet or How Not to Use XSLT', *University of California,Berkeley, School of Information Management and Systems* [Online] Available: http://www2.sims.berkeley.edu/academics/courses/is290-8/s04/lectures/5/dragons/allslides.html [accessed 8/13/2006]

Morabito, M (2003) 'CALCampus Origins', *Computer Assisted Learning Centre* [Online] Available: http://www.calcampus.com/calc.htm [accessed 3/9/2006]

Mozilla Development Centre (2007) 'Ajax: Getting Started' [Online] Available: http://developer.mozilla.org/en/docs/AJAX:Getting_Started [accessed 21/1/007]

Ogbuji, U (2001) 'Re: Rescuing XSLT from Niche Status' [Online] Available: http://www.biglist.com/lists/xsl-list/archives/200102/msg01119.html [10/8/2006]

Ostyn C, (2005) ' A Short History of SCORM' [Online] Available: http://www.ostyn.com/standards/docs/HistoryOfSCORM.htm [accessed 10/4/2006]

Reeves, T. C. (1993) ' Pseudoscience in computer-based instruction: The case of learner control research.' *Journal of Computer-Based Instruction, 20* (2), 39-46.

Shepherd, D (2001) 'Teach Yourself XML in 21 Days' 2nd ed. Indianapolis, SAMS.

Tennison, J (2004) 'Beginning XSLT', USA, Apress.

Van Meer, E (2003) 'PLATO: From Computer-Based Education to Corporate Social Responsibility' *Iterations An Interdisciplinary Journal of Software History* [Online] Available: http://www.cbi.umn.edu/iterations/vanmeer.html [accessed 3/9/2006]

Walsh, N & Muellner, L, (2004) 'DocBook: The Definitive Guide', O'Reilly & Associates

Williams, K (2002) ' XML for Data: XSL style sheets: push or pull?' [Online] Available: http://www-128.ibm.com/developerworks/xml/library/x-xdpshpul.html?open&l=976,t=gr [accessed 14/9/2006]