Final year project report	**Anirban Basu**	**Computer Systems Engineering (BEng)**

SCITECH, University of Sussex

ARCOLite – a visualisation system for viewing museum artefacts

Report Documentation Page

Report Documentation

Author	Anirban Basu
Course	Computer Systems Engineering (BEng)
Project first supervisor:	Dr M White
Project second supervisor:	Prof C R Chatwin
Period	October 6, 2003 – March 12, 2004
Distribution	BEng final year project report, dissertation
Department(s)	Department of Engineering and Design, Department of Informatics
School	SciTech, University of Sussex
Security	Internal
Page count including appendix	103

Glossary

Term	**Meaning**
ARCO	Augmented Representation of Cultural Objects
UML	Unified Modelling Language
IST	Information Society Technologies (http://www.cordis.lu/ist/)
X-VRML	XML-based dynamic VRML generation language
RDBMS	Relational Database Management System
XHTML	Extensible HyperText Markup Language
XDE	XML Data Exchange
ARIF	Augmented reality interface (client application)
CMAX	Content Management Application for XDELite
AXTE	ARCOLite XML Transformation Engine
MSXML	Microsoft XML technologies (http://msdn.microsoft.com/xml/)
JSP	Java Server Pages
W3C	World Wide Web Consortium (http://www.w3.org/)
MFC	Microsoft Foundation Classes
AMS	ARCO Metadata Schemas
VRML	Virtual Reality Modelling Language
DOM	Document Object Model
XSLT	Extensible Stylesheet Language Transformations
DTD	Document Type Definition
USB	Universal Serial Bus
RFC	Request for Comments

Technical terms in the context, which demand larger explanation and diagrams are discussed in the body of the main report. For a complete glossary of ARCO terms see ARCO-Glossary-R-1.0-280402.doc.

Summary

ARCO – Augmented Representation of Cultural Objects (http://www.arco-web.org/) is a EU IST Framework V funded research project aimed at providing museums with useful technologies for digitising, managing and presenting virtual museum artefacts in virtual cultural environments. ARCOLite is a derivative of ARCO focused on small and medium scale museums, reducing the total cost of ownership by eliminating expensive Oracle database and patented X-VRML technology. ARCOLite is self-sufficient and can communicate with external digital culture systems.

This final year project (*A visualisation system for viewing museum artefacts*) focuses on certain areas of the ARCOLite system. This report presents the overall concepts of the ARCOLite system and discusses in details the design and implementation of the underlying client-server architecture that forms the heart of ARCOLite by eliminating the equivalent Oracle and X-VRML architecture in ARCO. This report also describes the connectivity between an augmented reality client and the exhibition server.

Statement of originality

Date: Tuesday, 27 April 2004

I, hereby, certify that this technical report is a product of my own original work, unless otherwise stated. Any ideas or quotations from the work of other people published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline. This project has not been previously submitted, either in whole or in part, for a degree at University of Sussex or any other university.

This document template has been adapted from the ARCO BASE Template. All logos and trademarks are properties of their respective owners.

Anirban Basu

Year 3 in Computer Systems Engineering (BEng), University of Sussex

Table of Contents

List of Figures

Acknowledgements

I would like to take this opportunity to thank:

- Dr Martin White (first supervisor)
- Prof C R Chatwin (second supervisor)
- Mr Nicholaos Mourkoussis (DPhil student at Centre for VLSI and Computer Graphics, University of Sussex)
- Mr Fotis Liarokapis (DPhil student at Centre for VLSI and Computer Graphics, University of Sussex)
- Mr Miroslaw Stawniak (Marie Curie fellow at Centre for VLSI and Computer Graphics, University of Sussex)
- Ms Maria Sifniotis (DPhil student at Centre for VLSI and Computer Graphics, University of Sussex)
- Mr Alan Banks (Network administrator and technical support for Department of Engineering and Design)

Special thanks to my project first supervisor, Dr M White, for his useful advices and all the computational facilities he provided me with in his research lab at the Centre for VLSI and Computer Graphics.

And last but not the least, I would like to thank my parents, who have supported me emotionally and financially throughout the three years at University of Sussex, and encouraged me at every step to offer my very best.

1. Introduction

1.1 Background: ARCO – Augmented Representation of Cultural Objects

ARCO – Augmented Representation of Cultural Objects is a EU IST Framework V funded research project [1] aimed at providing museums with useful technologies for digitising, managing and presenting virtual museum artefacts in virtual cultural environments. The system architecture of ARCO is illustrated in Figure 1. This diagram is extracted from the research paper referenced in [10].

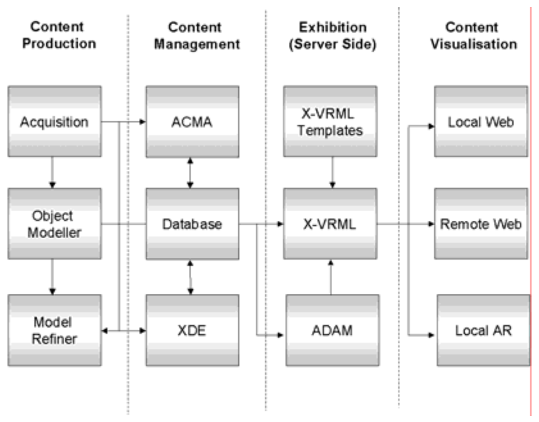


Figure 1. Conceptual architecture of the ARCO system

In the diagram, the *content management* and the *exhibition* components use database and a patented X-VRML technology [2]. ARCO uses Oracle 9i RDBMS. The X-VRML server works closely with a media server (called ADAM) and retrieves data from the database according to the user-agent (web browser or augmented reality client) requests. The X-VRML templates control the look-and-feel of the final output, which is streamed through a HTTP server (Apache in the current deployment) as pure XHTML content along with media (images, dynamic VRML, sound, etc.) content served through ADAM.

The content is visualised either in a standards compliant XHTML browser (Microsoft® Internet Explorer®, Mozilla, etc.) or in a specialised augmented reality client known as ARIF [3]. The ARIF application encapsulates a standards compliant XHTML browser. Virtual musuem exhibitions tailored for the web can be delivered to such a browser, while further interactivity is accomplished through the augmented reality table-top interface.

Two years down the line, it has become apparent to the designers of ARCO that the total cost of ownership of ARCO is too high due to the Oracle and X-VRML licensing issues. While there have been work done in the Poznan University of Economics (http://www.ae.poznan.pl/) to replace the database by an XML Data Exchange (XDE), ARCOLite is an even more lightweight derivative of ARCO that eliminates both the database and the X-VRML server and uses standard XML technologies for its data repository as well as content visualisation. This approach reduces the TCO making ARCOLite easily affordable by small and medium scale museums. In addition, ARCOLite demostrates a scalable interoperability prototype with any external digital culture system as well as learning scenario system [9].

Like ARIF in ARCO, an augmented reality client with an integrated standards compliant web browser also exists in ARCOLite. This is known as ARIFLite. However, work on ARIFLite beyond the level of connectivity to the ARCOLite client-server model, is not covered in this report because ARIFLite is not developed as a final year project but as a part of the ARCO research project.

1.2 System overview: ARCOLite and the scope of this final year project

The overall architecture of ARCOLite comprises of five conceptual parts or sub-systems.

1. Content production – a 3D authoring tool called Model Creator (MC) based on 3D Studio MAX. Content can also be produced (rather managed) by acquisition of other multimedia data images, video, audio, etc. Model Creator is out of the scope of this final year project.

2. Content management – a web-based content management system called Content Management Application for XDELite (CMAX), which will provide interoperability with external digital culture systems and will be able to package raw data from local or network storage for the ARCOLite data model. This final year project only uses a prototype of interoperability with external digital culture system (in our case, ARCO XDE) through an independent tool called the XDE-to-XDELite migrator. A full featured research on CMAX is outside the scope of this final year project but the migrator is discussed in detail.

3. XML data repository – an XML based repository called XDELite containing digital culture virtual exhibition data and presentation templates. XDELite is a collection of several XML schemas that map the underlying data-model used by ARCOLite. This forms an integral part of this final year project but is a collaborated work with Nicholaos Mourkoussis.

4. Exhibition server – a Java servlet based customisable XML transformation engine and a configurable media server, collectively known as ARCOLite XML Transformation Engine (AXTE) is deployed on a Apache Tomcat JSP/Servlet container. This forms another integral part of this final year project.

5. Content visualisation – this either refers to any standards compliant XHTML browser (for web content) or an integrated application encapsulating a web browser and augmented reality table-top environment called ARIFLite. This application is written in C++, using Microsoft Foundation Classes, MSXML, OpenVRML, ARToolkit [4]. Only the connectivity of this ARIFLite client to the exhibition server is a part of this final year project.

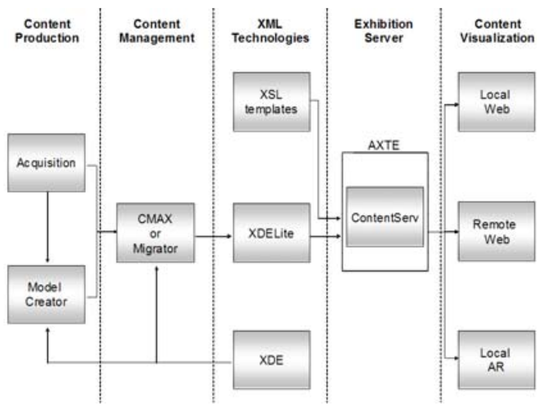


Figure 2. Conceptual architecture of ARCOLite system

Figure 2 illustrates the architecture of ARCOLite system. This diagram is extracted from the research paper [11]. At this juncture, it is important to note the exact scope of this final year project as illustrated above. To avoid any confusion, from this point forward, the noun *project* without any adjectives will refer to this final year project only unless explicitly mentioned otherwise. The exhibition server and the XML technologies form the core of this project. The migrator tool is an aid to data acquisition. On the other hand, the connectivity to augmented reality is an extra work done for this project.

Figure 3 shows an example deployment scenario of the conceptual architecture of ARCOLite in a museum. In this scenario, the content production and content management is done in-house within the museum networked environment. This digital content is stored and deployed on the exhibition server. Local or remote clients can then access the exhibition server to view the virtual exhibitions. Typical examples of local visualisation clients will be kiosk computers in museums that are capable of rendering web-based virtual exhibitions or augmented reality based interactive exhibitions. Remote presentations, over public wide area networks (like the Internet), can be rendered in web or augemented reality contexts.

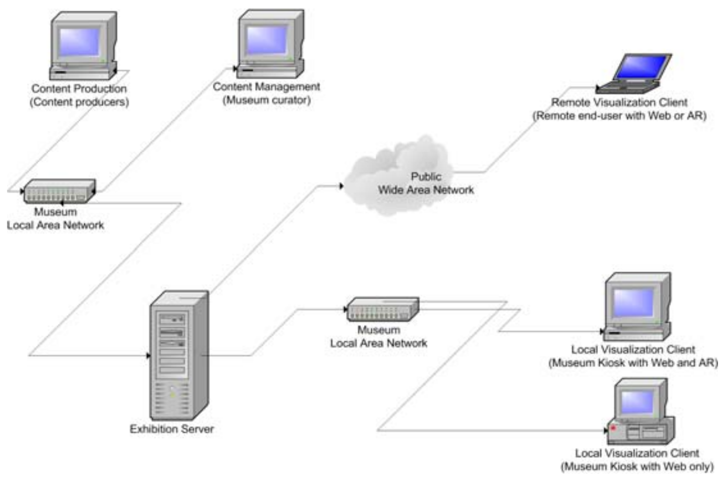

Content Production
(Content producers)
Content Management
(Museum curator)
Remote Visualization Client
(Remote end-user with Web or AR)
Museum
Local Area Network
Public
Wide Area Network
Museum
Local Area Network
Local Visualization Client
(Museum Kiosk with Web and AR)
Exhibition Server
Local Visualization Client
(Museum Kiosk with Web only)

Figure 3. An example networked deployment scenario of ARCOLite in a museum environment

1.3 Objectives

With this much of insight into the ARCOLite system overview in the light of the ARCO research project, I enlist the objectives of my final year project below.

1. Formally specify the system architecture and its user behaviour analysis.

2. Conceptualise the data model and implement the ARCOLite XML data repository.

3. Design and implement an interoperability tool – the migrator – for generating data for the ARCOLite XML data repository from an external system like ARCO.

4. Design and implement the exhibition server. Specify the working principle behind the web and augmented reality presentation templates. Deploy the exhibition server with sample presentation data and sample presentation templates.

5. Design and implement a communication interface between ARIFLite and the exhibition server.

This project spawns over several aspects and technologies of programming. The nature of the project requires enterprise class solutions to be deployed over short periods of time. The following development tools and programming technologies have been employed in this project.

- Software conceptualisation, design and analysis with Microsoft® Visio 2002 (http://www.microsoft.com/office/visio/).
- Data repository implementation using W3C XML (http://www.w3.org/XML/) and presentation style sheets using W3C XSL (http://www.w3.org/Style/XSL/) on Altova XMLSpy 4 (http://www.altova.com/).
- Rapid Application Development of servlets and code interfacing with XML using J2SE on Borland® JBuilder® 9 Enterprise (http://www.borland.com/jbuilder/). Use of Java for the server-side technologies also contributed to the architecture-neutral and platform-independent code.
- High-performance C and C++ code for augmented reality client application using MFC and several other libraries on Visual C++ 6 as a part of Microsoft® Visual Studio 6 Enterprise (http://www.microsoft.com/vstudio/).

2. System analysis

2.1 ARCOLite sub-systems and their components

Section 1.2 in the introduction can be summarised in the form of a UML component (deployment) diagram in Figure 4. In this deployment diagram, I have gathered the different components of the ARCOLite system and included them as parts of the aforementioned sub-systems. Italicized names of components indicate "abstract" or "external" components.

ARCO is classified as an external system to ARCOLite from the point-of-view of data flow. The only component of the ARCO system that is used in ARCOLite for exhibition data acquisition, through the migrator, is the XML Data Exchange (XDE).

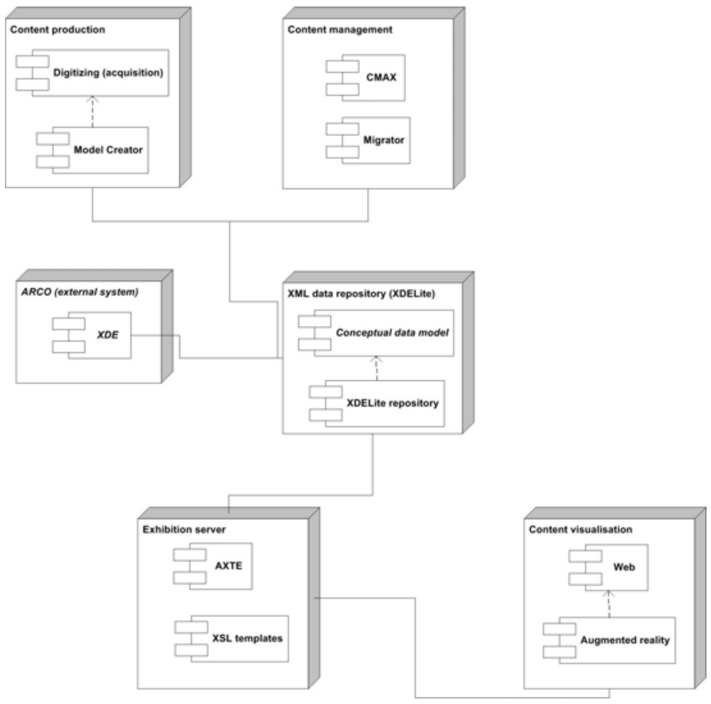


Figure 4. UML deployment diagram for the architecture of the ARCOLite system

The data model in the XML data repository sub-system is an abstract component. It is more of a concept than a physical entity. The schemas used in the XML data repository are responsible for implementing the data model along with some of the ARCO Metadata Schemas inherited from the ARCO data model [5].

In the following sub-section, I discuss about the user behaviour analysis for the entire ARCOLite system in the form of UML use-case. The ARCOLite sub-systems relevant to the final year project context are discussed in the next chapter.

2.2 User behaviour analysis

The conceptual user groups for the entire ARCOLite system can be summarised in the following list.

- Content Producer – an abstract user group in which users interact with the Content Production as well as the Content Management sub-system and any external digital culture system that has compatibility with ARCOLite. Content Producers acquire digitised virtual museum content from several sources. They also manage content by editing them from time to time.
- Exhibition Manager – an abstract user group in which users interact with the Content Management, the XML Data Repository and the Exhibition Server sub-systems. Exhibition Managers are a group of technical users who are responsible for using the content produced by Content Manager to create virtual museum exhibitions. Thus, Exhibition Managers can contain web designers. They are also responsible for deploying and testing virtual museum exhibitions.
- Exhibition Server – a non-abstract user and a sub-system in itself that is responsible for listening to client requests (from the Content Visualisation sub-system) and serving deployed data to the end-user.
- End-user – an abstract non-technical user group that interacts with the entire system to view virtual museum exhibitions. They interact with components in the Content Visualisation sub-system.

Figure 5 shows the behaviour of the aforementioned user groups in the form of a UML use-case diagram.

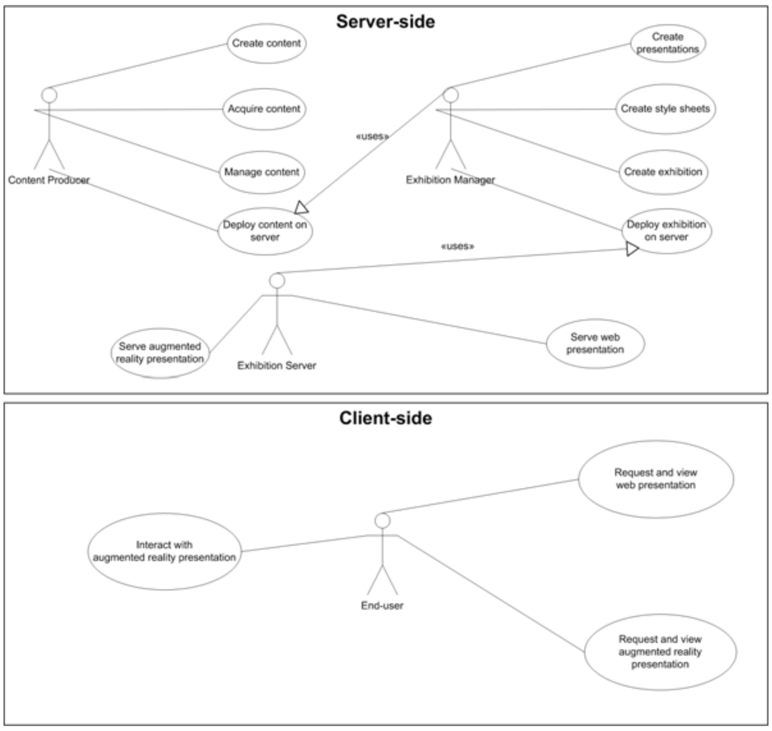


Figure 5. UML use-case diagram identifying the actors in the ARCOLite system

Having identified the user groups, I proceed into user behaviour analysis from the behaviours mentioned in the use-case diagram. The following table enlists the user behaviour of the abstract user group Content Producer.

Use-case	Description
Create content	The Content Producers have to create digitised content in some way. This is a generalised use-case to indicate the main activity of this group.

Acquire content	The Content Producers should be able to acquire content from external digital culture systems or acquire new content from specialised content creation equipment.
Manage content	Once obtained, the Content Producers must be able to edit the existing content by changing metadata, media files, etc.
Deploy content on server	The Content Producers should be able to deploy the acquired content on the exhibition server such that the Exhibition Managers can work on it to create virtual exhibitions.

The following table enlists the user behaviour of the abstract user group Exhibition Manager.

Use-case	**Description**
Create presentations	The Exhibition Managers should use the content deployed by the Content Producers to generate several virtual museum artefacts suitable for presentations.
Create style sheets	The Exhibition Managers should design the XSL style sheets to contain all the visual design as well as functionality required for dynamic processing of the virtual museum exhibitions.
Create exhibitions	Exhibition Managers should package several virtual museum artefacts to create virtual exhibitions.
Deploy exhibition on server	The Exhibition Managers should deploy virtual exhibitions on the server and test them to see that the Exhibition Server can serve the virtual exhibitions properly to the Content Visualisation sub-system.

The following table enlists the user behaviour of the non-abstract user Exhibition Server.

Use-case	**Description**
Serve web presentation	The Exhibition Server should be able to serve deployed virtual museum exhibitions to any standards compliant XHTML browser.
Serve augmented reality presentation	The Exhibition Server should be able to serve deployed virtual museum exhibitions to the specialised augmented reality client (ARIFLite) for interactive user experience.

The following table enlists the user behaviour of the abstract user group End-user.

Use-case	**Description**
Request and view web presentation	The End-users will be able to see virtual museum exhibitions on a standard web browser interface encapsulating virtual reality components like Cortona VRML client (http://www.parallelgraphics.com/products/cortona/).
Request and view augmented reality presentation	The End-users will be able to see virtual museum exhibitions on an augmented reality table-top in ARIFLite.
Interact with augmented	The End-users will be able to interact with virtual museum

reality presentation	exhibitions displayed on augmented reality table-top in ARIFLite with the help of special hardware equipment like Space Mouse (http://www.3dconnexion.com/). In future, ARIFLite will also implement interactive learning scenario context such as quizzes.

Having analysed the user behaviour in the ARCOLite system, I proceed with the discussion of each sub-system in ARCOLite in the form of overview, architecture, implementation and future directions (if relevant).

3. The ARCOLite sub-systems

3.1 XML data repository (XDELite): concept of the data model and the XDELite data repository

3.1.1 Overview

Before going into the detailed schema implementations of the different components of XDELite, it is important to identify these components and their requirement in the context. There are references to the ARCO XML Data Exchange (XDE) [5] from which the XDELite concepts are derived.

A representation of every virtual museum artefact is required. In XDE, each artefact is called a "cultural object" or CO. Cultural objects contain metadata (CO AMS) about artefacts and linked multimedia data as well as metadata (MO AMS and MOT AMS) for each such multimedia data (media objects). In XDELite, virtual museum artefact is represented as an "information resource" or IR. The generic naming convention suggests that the design of XDELite can be scaled to virtual representation of non-museum artefacts too. Unlike XDE, there is no concept of an acquired object or a refined object in XDELite. An IR consists of links to multimedia data (media objects) such as images, VRML, etc. Each such media object has metadata associated with it. Media objects that contain sub-media objects have sub-media object metadata associated with them. This concept is the same as in XDE. This defines the data model, which is a conceptual representation of a virtual museum artefact in XDELite. Figure 6 indicates the abstract component diagram for the conceptual data model. Concept of this data model is adapted from the research paper referenced in [13].

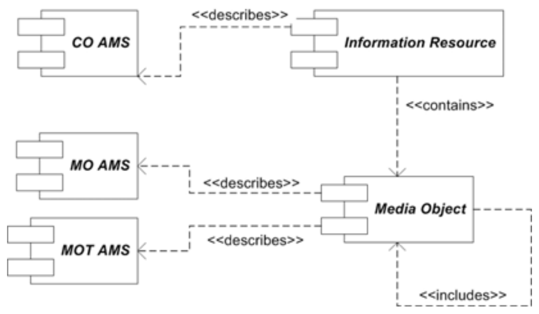


Figure 6. Abstract component diagram for the conceptual data model. The italicized text indicates "abstract" components.

A virtual exhibition is likely to contain more than one virtual museum artefact. Hence, there is a need for a packaging of information resources. In XDE, this is done through "cultural object folder" (COF). In XDELite, this is called PLITE – presentation lite or packaging lite. A PLITE instance consists of relative links to several IR instances and thus accomplishes the packaging of IR. PLITE also defines a name, an ID, and a description for each packaging instance.

These two identities (IR and PLITE) are practically sufficient for any single virtual museum exhibition. However, there is a requirement of several virtual exhibitions with several compatible web style sheets. In XDELite, this is achieved through something called IRF – Information Resource Folder. An IRF instance consists of links to several PLITE instances and several compatible XSL style sheets. Compatibility of style sheets is a design convention, and not a defined rule in XDELite.

The IR, PLITE and IRF are sufficient for any virtual museum exhibition over the web. For the augmented reality interface, there is still a need for another schema. In ARCO, there is no such schema. The ARIF application [3], in ARCO, is tailored to read the request parameters in the URL on the client-side before requesting data from the server. ARIF uses these parameters to determine the data that it needs to display on the augmented reality table-top. In ARCOLite, a more strict and neat solution is used by defining an entity called ARDATA. An ARDATA instance contains several AR objects, which are media objects that can be placed on augmented reality table-top. Each AR object defines a MIME type and an absolute HTTP link to the resource on the exhibition server. In addition, it can contain several AR sub-objects. Unlike IR, PLITE or IRF, ARDATA instances are generated on-the-fly only through specialised requests from the ARIFLite client.

3.1.2 Implementation

In the following sub-sections, the implementations of the schemas IR, PLITE, IRF and ARDATA are described. Code excerpts are shown to indicate implementation of each element. Where an element contains simple types or child elements, the code excerpts do not show all the code for the implementation but uses "…" to indicate missing code inside. Full schema implementation codes are enlisted in Appendix B – XML schemas.

3.1.2.1 IR schema

Figure 7 illustrates the complete schema for an information resource. It is implemented by eight elements and nine global elements.

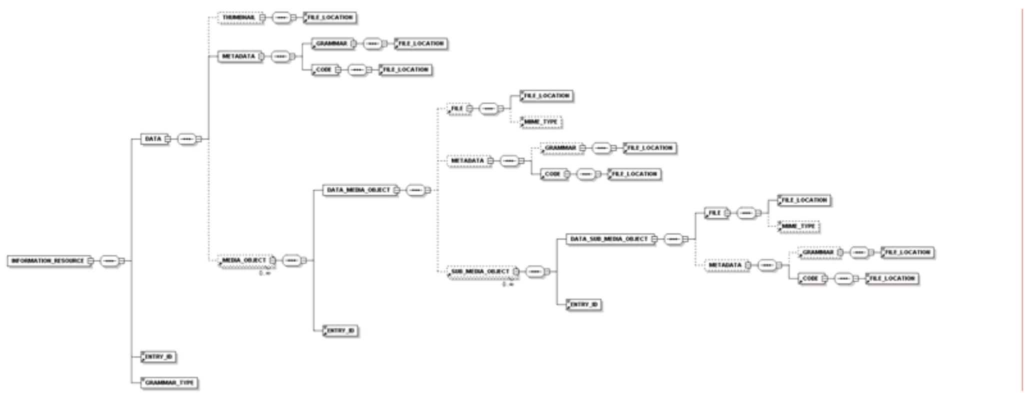


Figure 7. Schema for IR

Global element: MIME_TYPE

`MIME_TYPE` is a global element of type `xs:string`, which represents the MIME (Multipurpose Internet Mail Extensions) ([7] and [8]) type of media object or sub-media object. It is implemented by the following code.

```
<xs:element name="MIME_TYPE" type="xs:string"/>
```

Global element: ENTRY_ID

`ENTRY_ID` is a global element of type `xs:string`, which represents an alphanumeric identification code for an information resource, a media object or a sub-media object. It is implemented by the following code.

```
<xs:element name="ENTRY_ID" type="xs:string"/>
```

Global element: FILE_LOCATION

FILE_LOCATION is a global element of type xs:string, which represents a relative local path for a file identified by the FILE element or by the CODE element or by the GRAMMAR element or by the
THUMBNAIL element. It is implemented by the following code.

```
<xs:element name="FILE_LOCATION" type="xs:string"/>
```

Global element: CODE

CODE is a global element. It represents an instance of any of the metadata schemas mentioned in D11 and used in the ARCOLite system [5]. It has a mandatory child element that refers to the global element FILE_LOCATION, which points to the path of the metadata instance. It is implemented by the following code.

```
<xs:element name="CODE">...</xs:element>
```

Global element: GRAMMAR

GRAMMAR is a global element. It represents any of the metadata schemas mentioned in D11 and used in the ARCOLite system [5]. It has a mandatory child element that refers to the global element FILE_LOCATION, which points to the path of the metadata schema. It is implemented by the following code.

```
<xs:element name="GRAMMAR">...</xs:element>
```

Global element: FILE

FILE is a global element. It represents any media file (from a media object or sub-media object). It has one mandatory child element that refers to the global element FILE_LOCATION, which points to the path of the media file. The FILE element has another optional child element that refers to the global MIME_TYPE element, which represents the MIME type of the media file. It is implemented by the following code.

```
<xs:element name="FILE">...</xs:element>
```

Global element: MEDIA_OBJECT

MEDIA_OBJECT is a global element. It represents a media object with or without further sub-media objects. It has one mandatory child element called DATA_MEDIA_OBJECT. The MEDIA_OBJECT element has another mandatory child element that refers to the global ENTRY_ID element, which represents the identification code for the media object. It is implemented by the following code.

```
<xs:element name="MEDIA_OBJECT">...</xs:element>
```

Global element: SUB_MEDIA_OBJECT

SUB_MEDIA_OBJECT is a global element. It represents a sub-media object. It has one mandatory child element called DATA_SUB_MEDIA_OBJECT. The SUB_MEDIA_OBJECT element has another mandatory child element that refers to the global ENTRY_ID element, which represents the identification code for the sub-media object. It is implemented by the following code.

```
<xs:element name="SUB_MEDIA_OBJECT">...</xs:element>
```

Global element: INFORMATION_RESOURCE

INFORMATION_RESOURCE is the root element of the IR schema. It has one mandatory child element called DATA, which contains all the necessary information (in its child elements) about metadata, media objects and their sub-media objects. The INFORMATION_RESOURCE element has another mandatory child element that refers to the global ENTRY_ID element, which represents the identification code for the information resource. It also contains a third mandatory child element called GRAMMAR_TYPE. It is implemented by the following code.

```
<xs:element name="INFORMATION_RESOURCE">...</xs:element>
```

Element: GRAMMAR_TYPE

`GRAMMAR_TYPE` is a mandatory child element of `INFORMATION_RESOURCE`. It is a simple type with a restriction `xs:string` and an enumeration of two possible values `{Schema, DTD}`. This element specifies the type of grammar indicated by the global `GRAMMAR` element. For all purposes in ARCOLite, only `Schema` or XSD is used. This element may be dropped in future refinements of the IR schema. It is implemented by the following code.

```
<xs:element name="GRAMMAR_TYPE">...</xs:element>
```

Element: DATA

`DATA` is a mandatory child element of `INFORMATION_RESOURCE`. This element contains an optional `THUMBNAIL` element. It also contains a mandatory `METADATA` element. In addition, it contains `(0,*)` occurrences of references to the global `MEDIA_OBJECT` element. It is implemented by the following code.

```
<xs:element name="DATA">...</xs:element>
```

Element: THUMBNAIL

`THUMBNAIL` is an optional child element of `DATA`. It represents a thumbnail image for the information resource. It contains a mandatory child element that refers to the global `FILE_LOCATION` element, which points to the path of the thumbnail file. It is implemented by the following code.

```
<xs:element name="THUMBNAIL" minOccurs="0">...</xs:element>
```

Element: METADATA

`METADATA` is a mandatory child element of `DATA`. It represents the metadata grammar and its instance for the information resource. In ARCOLite, this refers to cultural object AMS metadata (section 2.3.1 in D11 [5]). It contains a mandatory child element that refers to the global `GRAMMAR` element and another mandatory child element that refers to the global `CODE` element. It is implemented by the following code.

```
<xs:element name="METADATA">...</xs:element>
```

Element: DATA_MEDIA_OBJECT

`DATA_MEDIA_OBJECT` is a mandatory child element of the global `MEDIA_OBJECT` element. This element contains an optional element referring to the global `FILE` element, which represents the media file. It also contains an optional `METADATA` element. In addition, it contains `(0,*)` occurrences of references to the global `SUB_MEDIA_OBJECT` element. It is implemented by the following code.

```
<xs:element name="DATA_MEDIA_OBJECT">...</xs:element>
```

Element: METADATA

`METADATA` is an optional child element of `DATA_MEDIA_OBJECT`. It represents the metadata grammar and its instance for the media object. In ARCOLite, this refers to media object AMS metadata (section 2.3.4 in D11 [5]). It contains an optional child element that refers to the global `GRAMMAR` element and a mandatory child element that refers to the global `CODE` element. It is implemented by the following code.

```
<xs:element name="METADATA" minOccurs="0">...</xs:element>
```

Element: DATA_SUB_MEDIA_OBJECT

`DATA_SUB_MEDIA_OBJECT` is a mandatory child element of the global `SUB_MEDIA_OBJECT` element. This element contains a mandatory element referring to the global `FILE` element, which represents the sub-media file. It also contains an optional `METADATA` element. It is implemented by the following code.

```
<xs:element name="DATA_SUB_MEDIA_OBJECT">...</xs:element>
```

Element: METADATA

`METADATA` is an optional child element of `DATA_SUB_MEDIA_OBJECT`. It represents the metadata grammar and its instance for the sub-media object. In ARCOLite, this refers to any sub-media object AMS metadata (MOT AMS in sections 2.3.5 through 2.3.10 in D11 [5]). It contains an optional child element that refers to the global `GRAMMAR` element and a mandatory child element that refers to the global `CODE` element. It is implemented by the following code.

```
<xs:element name="METADATA" minOccurs="0">...</xs:element>
```

3.1.2.2 PLITE schema

Figure 8 illustrates the schema for PLITE. It is implemented by five elements and two global elements, one of which is the root. The `BANNER` element may be dropped in any future refinement of the schema design.

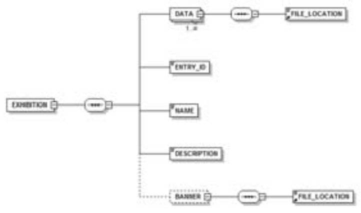


Figure 8. Schema for PLITE

Global element: FILE_LOCATION

`FILE_LOCATION` is a global element of type `xs:string`, which represents a local disk path for a file relative to the presentation root directory. It is implemented by the following code.

```
<xs:element name="FILE_LOCATION" type="xs:string"/>
```

Global element: EXHIBITION

`EXHIBITION` is the root element of the PLITE schema. It contains `(1,*)` occurrences of the `DATA` element, one `ENTRY_ID` element, one `NAME` element, one `DESCRIPTION` element and `(0,1)` occurrences of the `BANNER` element.

```
<xs:element name="EXHIBITION" type="xs:string">...</xs:element>
```

Element: DATA

`DATA` is a mandatory element that represents an IR instance and can have one or many occurrences representing the list of IR instances packaged by the PLITE instance. It has a mandatory child element that refers to the global element `FILE_LOCATION`, which represent the path of an IR instance file. The `DATA` element is implemented by the following code.

```
<xs:element name="DATA" maxOccurs="unbounded">...</xs:element>
```

Element: ENTRY_ID

`ENTRY_ID` is a mandatory element of type `xs:string`, which represents some kind of alphanumeric identification code for the PLITE instance. It is implemented by the following code.

```
<xs:element name="ENTRY_ID" type="xs:string"/>
```

Element: NAME

NAME is a mandatory element of type xs:string, which represents a relevant name for the exhibition package that is implemented by the PLITE instance. It is implemented by the following code.

```
<xs:element name="NAME" type="xs:string"/>
```

Element: DESCRIPTION

DESCRIPTION is a mandatory element of type xs:string, which represents a user-friendly description of the exhibition package. It is implemented by the following code.

```
<xs:element name="DESCRIPTION" type="xs:string"/>
```

Element: BANNER

BANNER is an optional element that represents the path for a banner image file that may be sometimes used by certain web style sheets (section 3.3.3.1). It has a mandatory child element that refers to the global element FILE_LOCATION, which represent the path of a banner image or media file. The BANNER element is implemented by the following code.

```
<xs:element name="BANNER" minOccurs="0">...</xs:element>
```

3.1.2.3 IRF schema

Figure 9 illustrates the schema for IRF. It is implemented by eight elements and one global element.

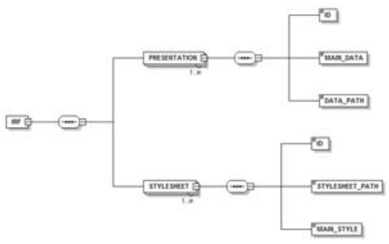


Figure 9. Schema for IRF

Global element: IRF

IRF is the root element of the IRF schema. It contains (1,*) occurrences of the PRESENTATION element and (1,*) occurrences of the STYLESHEET element.

```
<xs:element name="IRF">...</xs:element>
```

Element: PRESENTATION

PRESENTATION is a mandatory element that can have multiple occurrences. It contains one mandatory ID element, one mandatory MAIN_DATA element and one mandatory DATA_PATH element as children. The PRESENTATION element is implemented by the following code.

```
<xs:element name="PRESENTATION" maxOccurs="unbounded">...</xs:element>
```

Element: ID

ID is a mandatory child element of PRESENTATION. It is of type xs:int. It represents an integer identification code for the PRESENTATION. This identification code is used by the web style sheets (section 3.3.3.1) to select presentations. The ID element is implemented by the following code.

```
<xs:element name="ID" type="xs:int"/>
```

Element: MAIN_DATA

`MAIN_DATA` is a mandatory child element of `PRESENTATION`. It is of type `xs:string`. It represents the path for the PLITE instance that represents the presentation package, relative to the directory pointed to by the `DATA_PATH` element. The `MAIN_DATA` element is implemented by the following code.

```
<xs:element name="MAIN_DATA" type="xs:string"/>
```

Element: DATA_PATH

`DATA_PATH` is a mandatory child element of `PRESENTATION`. It is of type `xs:string`. It represents the path for the presentation directory that contains the PLITE instance pointed to by the `MAIN_DATA` element. The `DATA_PATH` element is implemented by the following code.

```
<xs:element name="DATA_PATH" type="xs:string"/>
```

Element: STYLESHEET

`STYLESHEET` is a mandatory element that can have multiple occurrences. It contains one mandatory `ID` element, one mandatory `STYLESHEET_PATH` element and one mandatory `MAIN_STYLE` element as children. The `STYLESHEET` element is implemented by the following code.

```
<xs:element name="STYLESHEET" maxOccurs="unbounded">...</xs:element>
```

Element: ID

`ID` is a mandatory child element of `STYLESHEET`. It is of type `xs:int`. It represents an integer identification code for the `STYLESHEET`. This identification code is used by the exhibition server (section 3.3) to select appropriate style sheets. The `ID` element is implemented by the following code.

```
<xs:element name="ID" type="xs:int"/>
```

Element: STYLESHEET_PATH

`STYLESHEET_PATH` is a mandatory child element of `STYLESHEET`. It is of type `xs:string`. It represents the path for the style sheet directory that contains the XSL file pointed to by the `MAIN_STYLE` element. The `STYLESHEET_PATH` element is implemented by the following code.

```
<xs:element name="STYLESHEET_PATH" type="xs:string"/>
```

Element: MAIN_STYLE

`MAIN_STYLE` is a mandatory child element of `STYLESHEET`. It is of type `xs:string`. It represents the path for the main XSL file that contains all the XSL templates for a particular web style sheet, relative to the directory pointed to by the `STYLESHEET_PATH` element. The `MAIN_STYLE` element is implemented by the following code.

```
<xs:element name="MAIN_STYLE" type="xs:string"/>
```

3.1.2.4 ARDATA schema

Figure 10 shows the schema for ARDATA. It is implemented two elements and three global elements.

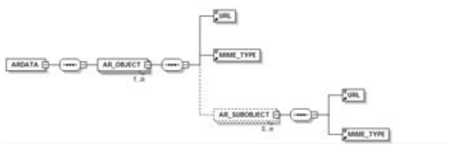


Figure 10. Schema for ARDATA

Global element: URL

URL is a global element of type xs:string, which represents an absolute HTTP URL to an AR_OBJECT or AR_SUBOBJECT on the exhibition server. It is implemented by the following code.

```
<xs:element name="URL" type="xs:string"/>
```

Global element: MIME_TYPE

MIME_TYPE is a global element of type xs:string, which represents the MIME (Multipurpose Internet Mail Extensions) type of an AR_OBJECT or AR_SUBOBJECT. It is implemented by the following code.

```
<xs:element name="MIME_TYPE" type="xs:string"/>
```

Global element: ARDATA

ARDATA is the root element of the ARDATA schema. It contains (1,*) occurrences of the AR_OBJECT element.

```
<xs:element name="ARDATA">...</xs:element>
```

Element: AR_OBJECT

AR_OBJECT is a mandatory element that can have multiple occurrences. It contains one mandatory element referring the global URL element that represents the absolute location of the AR object on the exhibition server; one mandatory element referring the global MIME_TYPE element that represents the MIME type of the AR object; and (0,*) occurrences of AR_SUBOBJECT element as children. The AR_OBJECT element is implemented by the following code.

```
<xs:element name="AR_OBJECT" maxOccurs="unbounded">...</xs:element>
```

Element: AR_SUBOBJECT

AR_SUBOBJECT is an optional child element of the AR_OBJECT element. It can have multiple occurrences. It contains one mandatory element referring the global URL element that represents the absolute location of the AR sub-object on the exhibition server; and one mandatory element referring the global MIME_TYPE element that represents the MIME type of the AR sub-object as children. The AR_SUBOBJECT element is implemented by the following code.

```
<xs:element name="AR_SUBOBJECT" minOccurs="0"
maxOccurs="unbounded">...</xs:element>
```

3.2 Content management: migrator as an independent tool

3.2.1 Overview

The entire content management sub-system of ARCOLite is beyond the scope of this project. It is also under research and development at this stage. To populate the XML data repository with sample data so that the exhibition server can work on the data, I have designed an independent migration tool that operates on data exported from an external digital culture system (ARCO in this case) and populates the XML data repository of ARCOLite. The migrator tool is a research-in-progress tool developed to illustrate the interoperability of ARCOLite with another external digital culture system ARCO. It is important to note that even if many concepts in ARCOLite have been derived from ARCO, the ARCOLite still sees ARCO as an external system. The migrator tool is, at present, a standalone Java application natively compiled on Windows platform. In the course of research of ARCOLite, the internal libraries of the migrator tool will be reused in the Content Management Application for XDELite (CMAX).

3.2.2 Architecture

The migrator tool has four main components. This is illustrated in Figure 11.

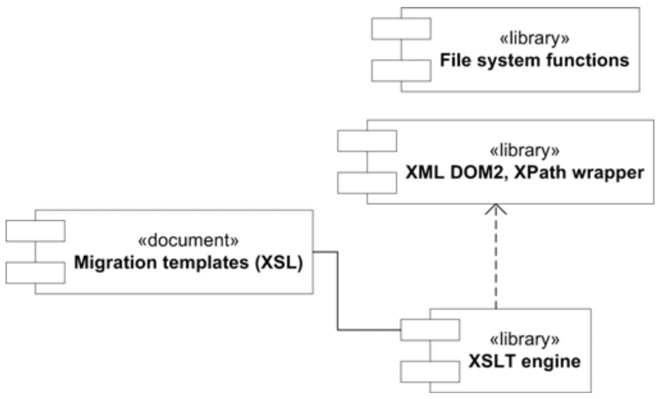


Figure 11. Component diagram for the migrator

- *File system functions* is a library that encapsulates file system functionality required by the migrator like reading text files (VRML), creating, deleting, renaming files or directories, etc.
- *XML DOM2, XPath wrapper* is a library that packages the basic functionality of W3C DOM2 I/O and XPath inside the migrator. This is used to read XML files into DOM2 tree structures and to efficiently search such DOM2 trees through XPath.
- *XSLT engine* is a library that is used by the migrator to apply XML transformation templates on XML data files along with custom parameters passed as a hash table.
- *Migration templates (XSL)* are two XSL style sheet documents. One of these style sheets is used to transform each cultural object in an XDE instance into an information resource. The other style sheet is used to package all the generated IR instances into one PLITE instance.

The functionality of migrator is illustrated in the form of an UML state chart in Figure 12.

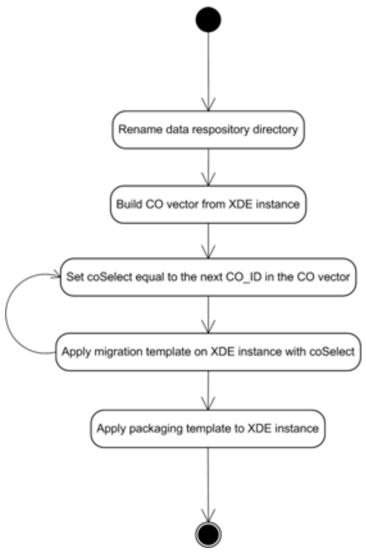

Rename data respository directory
Build CO vector from XDE instance
Set coSelect equal to the next CO_ID in the CO vector
Apply migration template on XDE instance with coSelect
Apply packaging template to XDE instance

Figure 12. UML state chart for the migrator

The migrator uses file system functions to rename the XDELite data repository directory from `arco_data` to `ARCO_DATA`. This is required because of the way the texture URLs are referenced inside VRML media files at a later stage and also because the Apache Tomcat JSP/Servlet container used in the exhibition server is case-sensitive.

Next, the migrator uses W3C DOM2 library functions to load the XDE instance into a W3C DOM2 tree structure. It then uses the Apache XPath API to build a vector of `//ARCO_DATA/DYNAMIC_DATA/CULTURAL_OBJECTS/CO` nodes containing their `CO_ID` attributes. At this stage, the migrator knows exactly which cultural objects have to be converted to information resources. The XDE migrator template expects a parameter called `coSelect`, which is the ID of a cultural object that needs to be converted into an information resource. Thus, iterating through each CO_ID of the CO vector, the migrator calls the XSLT engine with the XDE instance as the input XML, the migrator template as the style sheet, and a hash table containing the ID of the selected cultural object. The output file name is generated on-the-fly using the ID of the selected cultural object and some pre-specified text. This apparently gives a functional impression that the migrator outputs several IR instances out of one XDE instance although in essence, it outputs one IR instance in each iteration as it goes through the CO vector.

When the generation of all these IR instances is complete, the migrator calls XSLT engine with the XDE instance, the PLITE generation template and with certain parameters that specify the IR file naming convention used. The XSL template reads all the `//ARCO_DATA/DYNAMIC_DATA/CULTURAL_OBJECTS/CO` nodes and generates equivalent nodes in the PLITE instance having values concatenated with the passed parameters to point exactly to the IR instances generated so far.

3.2.3 Implementation and results

Figure 13 shows the UML class diagram of the main executable class (`BatchProcess`) of the migrator and its dependencies on other classes in the Java API (`java.lang`, `java.io` and `java.util` packages) as well as custom classes in the `migrator` package.

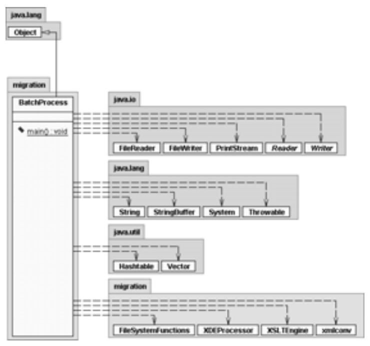
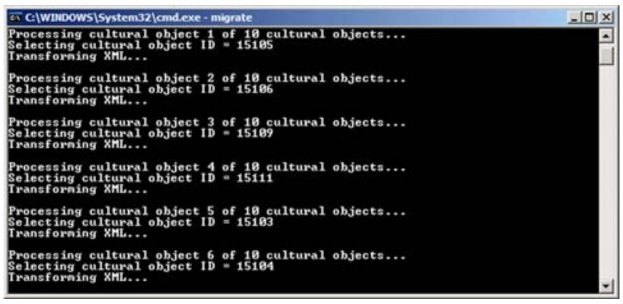


Figure 13. UML class diagram of the migrator main executable class

From the point-of-view of functionality, the migrator tool is a command line application that applies a XSL migration template on a XDE instance to generate several IR instances and finally package them into one PLITE instance through another XSL template. The migrator does not generate an IRF instance. This is hand-coded at the time of deployment in the exhibition server. The migrator does not also change relative texture URLs inside VRML media files in the XDELite data repository. This is also hand-coded at the time of deployment, especially because there are version compatibility issues of different XDE schemas. Thus the migrator tool is a fast mechanism for data acquisition for the exhibition server but not a complete tool in itself. Figure 14 shows the migrator tool at work from a Windows XP command line. The migrator has been compiled into a native executable through Borland JBuilder 9 Enterprise.



```
C:\WINDOWS\System32\cmd.exe - migrate
Processing cultural object 1 of 10 cultural objects...
Selecting cultural object ID = 15105
Transforming XML...

Processing cultural object 2 of 10 cultural objects...
Selecting cultural object ID = 15106
Transforming XML...

Processing cultural object 3 of 10 cultural objects...
Selecting cultural object ID = 15109
Transforming XML...

Processing cultural object 4 of 10 cultural objects...
Selecting cultural object ID = 15111
Transforming XML...

Processing cultural object 5 of 10 cultural objects...
Selecting cultural object ID = 15103
Transforming XML...

Processing cultural object 6 of 10 cultural objects...
Selecting cultural object ID = 15104
Transforming XML...
```

Figure 14. Screenshot of the migrator

3.2.4 Experimentation and future work

At this level, the migrator provides functionalities sufficient enough to acquire data for the exhibition server. However, I experimented further both from a conceptual point-of-view and an implementation point-of-view towards migrating textures and speeding up respectively.

The textures references in VRML files as exported from the Oracle 9i database with the XDE instance in ARCO have changed recently. When I first started to write the migrator, the texture references contained URLs like `getmo?id=XXXXX`, where `XXXXX` specified an ID for the media object in the XDE instance. Conceptually, one could look up this ID in the generated IR instance that contains references to the VRML file and then retrieve the actual filename in which this file is saved in the XDELite data repository. This was the technique used for hand-coding the appropriate texture URLs for the exhibition server. However, the more recent version of XDE specified file names for texture URLs as exported from authoring tools like 3D Studio MAX. This approach certainly makes it a bit more complicated to figure out the actual file names for these texture files in the XDELite data repository. I envisaged a concept of using a hash table of hash tables that can be generated by reading the XDE instance on its own. One hash table (say, A) contains appropriate conversions between the exported file names and actual file names of all media objects for one cultural object. The second hash table (say, B) contains several such “A” hash tables corresponding to each cultural object and are identified by their IDs. The migrator could then use this hash table combination for very fast lookup and change texture URLs inside the VRML files without even having to refer back to the XDE instance or the IR instances for each URL. This idea has been developed in its conceptual level only and never been implemented in the migrator.

By the time, the current migrator was able to convert XDE instances to IR instances, I thought of experimenting with performance issues. The current migrator might be a bit slow because it runs indirectly on the Java Virtual Machine even if it is some kind of a native version from JBuilder Enterprise 9. Therefore, I thought of implementing the migrator partially (just generate the IR instances) to prove the concept in C++ using Microsoft Foundation Classes and Microsoft XML parser library. As a result, I implemented a wrapper class around MSXML4 to serve my purpose. This class is later used in the ARIFLite application for connectivity between the augmented reality system and the exhibition server. It contains XSLT processing capabilities. Given the migration procedure described earlier in section 3.2.2, it is apparent that even though the XDE instance is required in each XML transformation that generates an IR instance, it is required only for data input and during these transformations the data inside the XDE instance do not change. Thus, I thought of parallelising the XML transformation tasks using the XDE instance as a shared resource without any mutex lock on the file itself. The UML state chart for a concurrent migrator is illustrated in Figure 15. Instead of a loop (as in Figure 12), the actual CO to IR migration procedure is concurrent.

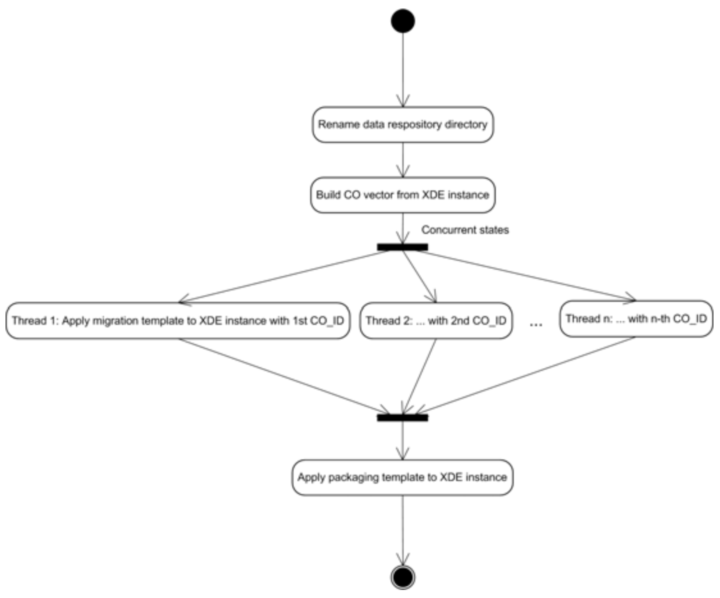

Rename data respository directory
Build CO vector from XDE instance
Concurrent states
Thread 1: Apply migration template to XDE instance with 1st CO_ID
Thread 2: ... with 2nd CO_ID
...
Thread n: ... with n-th CO_ID
Apply packaging template to XDE instance

Figure 15. UML state chart for a concurrent migrator

I used the worker thread concept in MFC to write a multithreaded partial migrator and the speed-up was manifold. The application creates *n* worker threads for *n* cultural objects and processes each cultural object concurrently. Further to this, I tested this application on a multiprocessor machine (running two Intel® Xeon™ processors at 2.4GHz each) and the operating system (Windows XP) actually runs those threads on two processors contributing to an even further performance increase. Although this concurrency was just at the experimentation level, future research work on the Content Management Application for XDELite may use a built-in concurrent migrator. Even if that will be in Java, with the recent JDK, Java threads are linked to kernel threads, which means multiprocessor environments will certainly contribute to performance enhancement. Java also provides very safe multithreading libraries because each object in Java is a monitor in itself.

3.3 Exhibition server: servlet architecture and presentation templates

3.3.1 Overview

The fundamental functional unit of this project is the exhibition server. The exhibition server is a Java servlet architecture known as AXTE (ARCOLite XML Transformation Engine), code-named ContentServ, deployed on Apache Tomcat JSP/Servlet container along with the XDELite repositories select for exhibits. The exhibition server is responsible for serving any content from the XDELite repository to any visualisation client (web or augmented reality) on the client end.

At the start of the project, AXTE was implemented in two different Java servlets – XView and MediaView. XView was responsible for serving dynamic XHTML content generated from the XDELite data repository and presented according to pre-determined templates. MediaView, on the other hand, was responsible for serving media content (mostly binary data) from the XDELite data repository. At this stage, there were no concepts of multiple presentation and multiple compatible presentation templates. In the later half of the project work (January 2004 onwards), the needs for multiple presentations and multiple compatible presentation templates were felt and the concept of IRF (section 3.1) was introduced. Immediately, certain limitations in the servlets were noticed. One fundamental issue arose regarding texture URLs in VRML files. So far, the textures were served through MediaView and the texture URLs used to look like `MediaView?filename=ARCO_DATA//MO_BINARY_DATA_XXXXX`. In this approach, MediaView always assumed that there was only one ARCO_DATA directory relative to the root of the servlet context path. However, with multiple presentation data there are several ARCO_DATA directories in several presentation sub-directories relative to the root of the servlet context path. Also it should be possible to port a VRML file from one presentation to the other without having to change the texture URLs. This leads to an idea of letting the servlet serve the media content from the context path of the *current presentation*, where *current presentation* dynamically changes depending on which presentation the user requests. Thus, texture URLs now look like `ContentServ?modeID=2&filename=ARCO_DATA//MO_BINARY_DATA_XXXXX`. This does not specify which presentation directory the servlet should look into but it is assumed that the servlet will look into the presentation directory of the *current presentation*.

3.3.2 Architecture

To implement the idea of *current presentation* and eventually *current style sheet* discussed in the above section, I employed persistent storage of presentation and style sheet information using servlet sessions. Since both the XML transformation servlet and the media servlet require storing the same *current presentation* and *current style sheet*, the former XView and MediaView were merged into one single servlet called ContentServ having four non-concurrent mutually exclusive states of operation. These states or modes are:

- MODE_XFORM: This is the XML transformation mode that uses servlet sessions to store persistent information about presentation and style sheets.
- MODE_BINMED: This is the mode for serving binary stream of media content using servlet sessions to correctly identify the presentation in use. Although media content like VRML is actually character data, the binary mode of this servlet actually treats all input data as binary stream and outputs them to the `ServletOutputStream` object to ensure that there is no character conversion.
- MODE_BINTPL: This is the mode for serving binary stream of media files associated with the templates (like static images, static cascaded style sheets, static scripts, static XHTML pages, etc.). This mode also uses servlet sessions to identify the current presentation style sheet in use.
- MODE_XFORM_PLAIN: This is another XML transformation mode that does not use any servlet session information to identify presentations or style sheets but treat all links to the source XML and the given XSL relative to the root of the servlet context path.

These modes change the operations of the servlet. The modes are specified by the integer request parameter `modeID`. Figure 16 illustrates the four non-concurrent mutually exclusive states of AXTE in the form of a UML state chart. When the servlet is invoked for the first time in any client session, the user has to specify the presentation, the presentation template, and the IRF instance to look into through request parameters `pID`, `sID` and `irfID` respectively. The current deployment uses only one instance of IRF but having it specified by a request parameter makes it possible to have more than one IRF instances in the root of the servlet context path.

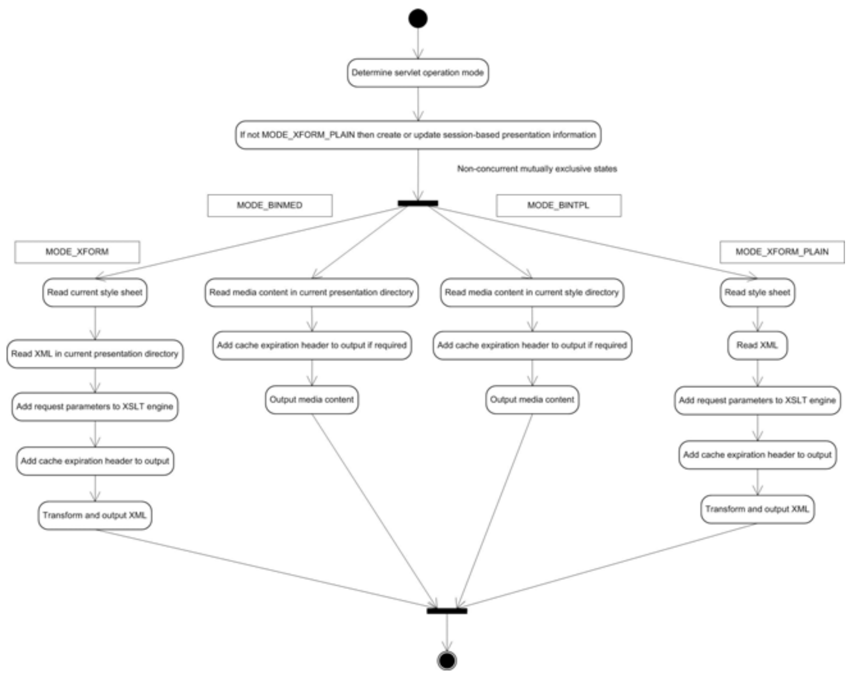

Determine servlet operation mode
If not MODE_XFORM_PLAIN then create or update session-based presentation information
Non-concurrent mutually exclusive states
MODE_BINMED
MODE_BINTPL
MODE_XFORM
MODE_XFORM_PLAIN
Read current style sheet
Read XML in current presentation directory
Add request parameters to XSLT engine
Add cache expiration header to output
Transform and output XML
Read media content in current presentation directory
Add cache expiration header to output if required
Output media content
Read media content in current style directory
Add cache expiration header to output if required
Output media content
Read style sheet
Read XML
Add request parameters to XSLT engine
Add cache expiration header to output
Transform and output XML

Figure 16. UML state chart for AXTE

There is something common to the modes MODE_XFORM, MODE_BINMED and MODE_BINTPL: the concept of servlet session management for persistent storage of presentation and style sheet information. Therefore, when the servlet is invoked and as long as the specified mode of operation is not MODE_XFORM_PLAIN, the following session management takes place before the servlet works according to the specified mode.

The servlet has to read in the relative path of the requested presentation and the relative path of the requested style sheet from the specified IRF instance. The servlet first checks if there is a session variable VAR_PRESENTATION (please refer to `ContentServ.java` in the source code) amongst the session variables. If it does not exist, then the servlet knows that either it is being invoked the first time in the client session or the previous session has expired, which means it has to update the VAR_PRESENTATION again. Irrespective of the result of this check, the servlet calls a method called `processRequestParameters(HttpServletRequest request)`. This method checks for session variable VAR_PRESENTATION availability. If it has expired or does not exist, then it checks if the given parameters are sufficient to read in the data required for VAR_PRESENTATION, which means `pID`, `sID` and `irfID` must be specified. On the other hand if VAR_PRESENTATION does exist then the method updates it only if the requested parameters indicate that an update is necessary. If an update is not necessary, the presentation and style sheet related information is maintained in the session so that the servlet can use it while operating in different modes. VAR_PRESENTATION is actually an object of class `Presentation` (please refer to `Presentation.java` in the source code) that is used to store the paths for the selected presentation and the selected style sheet from the IRF instance. The `Presentation` class is very much like a custom data structure that can be defined using `struct` keyword in C or C++. The following code snippet shows the declaration of this data structure in Java.

```
class Presentation
{
  protected int pID, sID; //the requested presentation ID and style sheet ID
  protected String path; //this is presentation directory
  protected String mainData; //PLITE instance
  protected String stylesheetPath; //path of the stylesheet directory
  protected String styleSheet; //associated main style sheet
}
```

There is a class called `PresentationReader`, which helps the servlet to read an IRF instance using an underlying XML wrapper architecture similar to that used in the migrator. It uses W3C DOM2 to read the presentation and style sheet path information.

If the requested mode of operation is MODE_XFORM_PLAIN, the servlet does not go through the session management stage explained above. In the following sub-sections, I discuss the modes of operation in more details.

3.3.2.1 Transformation modes: MODE_XFORM and MODE_XFORM_PLAIN

Both these modes use an XSLT processor at heart that acts on a XML instance file and applies a XSL template to it. These modes output character data stream. For maximum flexibility at the XSL template design level, the servlet adds all the request parameters to the XSLT engine, many of which are used by the templates. Two parameters, `source` and `style` are special. When in MODE_XFORM, these are usually `currentPresentation.mainData` and `currentPresentation.styleSheet` respectively where `currentPresentation` is an object of class `Presentation` and is maintained in VAR_PRESENTATION session variable. However, if the template requires then it can override the values of these parameters by explicitly specifying them in the request parameters. When in MODE_XFORM_PLAIN, the `source` and the `style` parameters must be explicitly specified by the user. The servlet also adds additional parameters like host name, host port, servlet path, and servlet context path root because these are used by the XSL templates for dynamically generating certain links. This is discussed in further details in the templates section. Before transforming XML calls a method `addCacheExpirationModel(HttpServletResponse response)`, which adds HTTP response headers to the servlet output to make sure that the browser does not cache the output. Caching of output causes problems while working with MODE_XFORM because sometimes certain URLs are displayed from the browser cache and the servlet gets confused about VAR_PRESENTATION session variable when certain other URLs are requested. The added cache expiration headers conform to sections 13 and 14 of HTTP 1.1 RFC 2068 [6]. This cache expiration model works through HTTP proxies too. Finally, the output of the XML transformation is sent to the servlet output. This output is usually XHTML, but can also be XML especially when serving requests for the augmented reality client – ARIFLite.

3.3.2.2 Binary content serve modes: MODE_BINMED and MODE_BINTPL

Both these modes are responsible for reading in files, specified by the `filename` request parameter, using binary file I/O on the exhibition server end and sending the data into the servlet output stream. When in MODE_BINMED, the presentation context path (`currentPresentation.path`) is used to resolve all relative file paths specified by the `filename` parameter. In MODE_BINTPL, the style sheet context path (`currentPresentation.stylesheetPath`) is used to resolve the relative file paths. Like MODE_XFORM and MODE_XFORM_PLAIN, the binary modes also have the same cache control technique. MODE_BINTPL adds cache expiration headers to make sure that the browser always gets the data from the servlet and nothing is cached. However, in the MODE_BINMED, a special request parameter – `cache` – can be specified to let the browser cache the content. Any value for this parameter will let the browser cache the media content. By default, this parameter is not present and hence the browser cannot cache any media content. Caching of media content may sometimes be necessary when switching rapidly between browser and augmented reality mode in the ARIFLite application and displaying objects in augmented reality that have been displayed previously in the same session. This is used to reduce network overhead (important for large media content over slow networks) and provide faster response to the user.

3.3.3 Implementation and results

Figure 17 shows the UML class diagram of the main servlet class (`ContentServ`) of AXTE and its dependencies on several classes in the Java API as well as custom classes in the `axte` package.

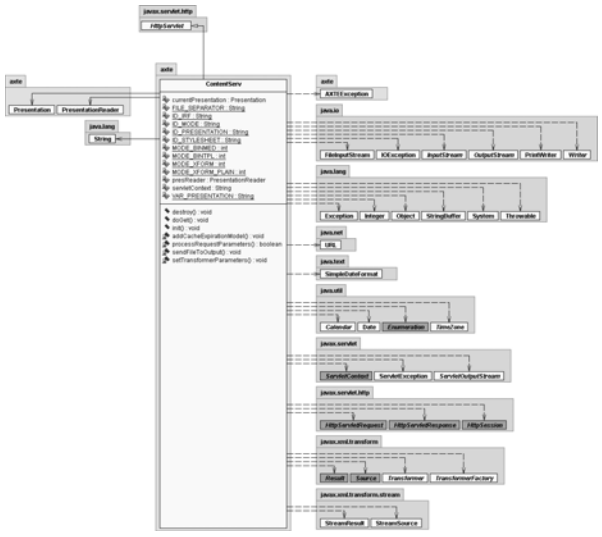


Figure 17. UML class diagram of the AXTE ContentServ servlet class

It is apparent from the above discussion that the servlet provides a framework to perform all the tasks necessary to render dynamic content on the web and augmented reality interfaces. However, the servlet itself does not govern the dynamic output. It is the presentation templates that do so. At this point, I state that when the term "presentation template" or "template" or "presentation style sheet" or "style sheet" is used with a "web" qualifier then it refers to the XSL templates that output pure XHTML content. When the qualifier is "augmented reality" or "AR" then it refers to XSL templates that output pure XML content conforming to the ARDATA schema.

3.3.3.1 Presentation templates – web style sheets

The presentation templates are very flexible and extensible in the way they are designed, mainly because of the use of XSL. The look-and-feel is largely governed by the static XHTML code and the specific visual requirements of that presentation. However, there are certain methodologies used in each web template to render dynamic content. For example, the `<xsl:template match="/">` node in a XSL style sheet is treated very similar to the concept of a `public static void main(String [] args)` in Java. This means that the XSL processor starts parsing the template from this entry-point. The rest of the functionality in the style sheet is modularized into several templates like `<xsl:template name="mainframe">`, which outputs the main frameset of the dynamic web pages. Each such modular template is called with or without parameters depending on custom request parameters passed by the servlet to the XSLT engine. For the sake of clarity, I will call the `<xsl:template match="/">` as the "main template" or the "entry-point template".

The web templates are designed to generate pure XHTML content. Most of the design elements of the web pages are irrelevant to the XSLT engine and they are designed in HTML editors like Macromedia® Dreamweaver. One fundamental concept common to all the web templates I have used is frameset. A single web page cannot call the servlet recursively with different parameters to generate more dynamic hypertext content inside it unless using framesets, or other client-side scripting to display hypertext content inside layers, etc. I used framesets as a solution.

The web templates use several parameters, many of which are dependent on specific needs. There are certain parameters common to all like `source`, `style`, `pID`, `sID`, `irfID` and environment parameters like `contextPath`, `servletPath`, `host`, `port`, etc. There is a parameter called `action`, which is used to select the templates inside one style sheet file in very much the same way as `arMode` parameter is used in the augmented reality templates. Apart from this, there are other parameters that enable the implementation of a paging system to browse the entire catalogue of virtual objects in a presentation.

The web style sheets can vary widely depending on requirements. I explain certain functionality issues in an example web style sheet for the Fishbourne Roman Palace presentation (`frp.xsl`). The visual design of this template has been done by Maria Sifniotis. A sample output of this style sheet is shown in Figure 19.

There are seven modular templates in this web style sheet. These are discussed as follows.

- Template – main frameset (`<xsl:template name="mainframe">`): This template sets up the main frameset and centers the nested frameset. It also generates the URL to the content of the nested frameset dynamically.
- Template – nested frameset (`<xsl:template name="nestedframe">`): This template sets up the nested frameset. It generates the URLs to the contents of the header page, the footer page and the introductory page dynamically.
- Template – header page (`<xsl:template name="header">`): This template dynamically generates the name of the selected presentation from the appropriate PLITE instance and adds the design elements dynamically from the selected style sheet directory.
- Template – footer page (`<xsl:template name="footer">`): This template dynamically generates links to other presentations in the same level (level 1) or the parent level (level 0), which is governed by an IRF instance. It also adds design elements and copyright information.
- Template – dynamic content selection (`<xsl:template name="content">`): This template does not generate any visual content but is responsible for intercepting user requests to select the appropriate content template, such as the introductory level or information resource level or media level (in some other style sheets), etc.
- Template – introductory content page, level 1 (`<xsl:template name="contentIntro">`): This template is responsible for outputting dynamic content to generate a list of information resources (IR instances) available in a presentation package (PLITE instance). It adds a link to the appropriate augmented reality style sheet that is able to generate a list of all AR_OBJECT entries in the selected presentation. This is required for the “magic book” scenario. This template provides the user with a user-friendly catalogue-like structure with each item indicated by a thumbnail image and a link to the appropriate information resource. The thumbnail, the name, the link, etc. are dynamically generated from the appropriate information resource instances. This catalogue or list is broken down into several pages so that one page is not cluttered with several items. The paging system is dynamic and is controlled by global variables `page_size` (number of items in a page) and `focp` (index of the first item on current page). The following code excerpt shows the implementation of the dynamic paging system.

```
	<xsl:if test="floor(($focp - 1) div $page_size) &gt; 0">

	<a>

	<xsl:attribute name="href">ContentServ?modeID=1&amp;source=<xsl:value-of
select="$source"/>&amp;action=content&amp;isIntro=<xsl:value-of
select="$isIntro"/>&amp;focp=<xsl:value-of select="$focp –
$page_size"/>&amp;pID=<xsl:value-of select="$pID"/>&amp;sID=<xsl:value-of
select="$sID"/>&amp;irfID=<xsl:value-of select="$irfID"/></xsl:attribute>

	<img src="ContentServ?modeID=3&amp;filename=images/left.gif" border="0">
```

```
             <xsl:attribute name="alt">Previous <xsl:value-of
select="$page_size"/> objects</xsl:attribute>

      </img>

      </a>

      </xsl:if>
                                                            </td>
                                                            <td
align="right" valign="middle">

      <xsl:if test="floor(($total_nodes - $focp) div $page_size) &gt; 0">

      <a>

      <xsl:attribute name="href">ContentServ?modeID=1&amp;source=<xsl:value-of
select="$source"/>&amp;action=content&amp;isIntro=<xsl:value-of
select="$isIntro"/>&amp;focp=<xsl:value-of select="$focp +
$page_size"/>&amp;pID=<xsl:value-of select="$pID"/>&amp;sID=<xsl:value-of
select="$sID"/>&amp;irfID=<xsl:value-of select="$irfID"/></xsl:attribute>

      <img src="ContentServ?modeID=3&amp;filename=images/right.gif"
border="0">

             <xsl:if test="($total_nodes - ($focp + $page_size)) &gt;=
$page_size">

                   <xsl:attribute name="alt">Next <xsl:value-of
select="$page_size"/> objects</xsl:attribute>

             </xsl:if>

             <xsl:if test="($total_nodes - ($focp + $page_size)) &lt;
$page_size">

                   <xsl:attribute name="alt">Next <xsl:value-of
select="$total_nodes mod $page_size"/> objects</xsl:attribute>

             </xsl:if>

      </img>

      </a>

      </xsl:if>
```

- Template – information resource content page, level 2 (`<xsl:template name="contentIR">`): This template is responsible for dynamic content generation for a selected information resource. This involves displaying media objects (usually image and VRML) along with metadata for an information resource. In this process, the template actually reads the metadata schema too to generate tool-tips from the application profile information [12].

3.3.3.2 Presentation templates – augmented reality style sheet

At present there is only one type of augmented reality style sheet. The augmented reality client (ARIFLite) is capable of displaying only VRML objects on markers at the moment. Hence the augmented reality templates are configured to make the servlet serve VRML objects (and their texture files) to ARIFLite.

The entry-point in the augmented reality style sheet is the `<xsl:template match="/">` node. The style sheet uses some global parameters, most of which are “environment variables”, such as `host`, `port`, `servletPath`, etc. The `pID`, `sID` and `irfID` parameters are used to explicitly specify the presentation and style sheet that is calling the augmented reality templates. This is important if the augmented reality client does not connect to the exhibition server using the same HTTP session as the embedded browser from which the augmented reality client is triggered. The `source` parameter is used to explicitly specify on which XML file the templates must be applied. There is another parameter called `arMode`, which can have three values – `listP`, `listI` and `selectI`. The `listP` mode outputs each and every VRML object (and their textures) from each and every IR instance contained in the PLITE instance pointed to by the `source` parameter. A typical use of this is in the "magic book" scenario. The `listI` mode outputs each and every VRML object (and their textures) in an IR instance pointed to by the `source` parameter. The `selectI` mode outputs only one media object (not just VRML) from an IR instance specified by the `source` parameter. The media object to be outputted is selected by another request parameter called `arSelect`.

There are two modular templates `<xsl:template name="listIR">` and `<xsl:template name="listPLITE">`, which implement the above functionality. The main template determines which modular template to call depending on the `arMode` parameter. The `listPLITE` template calls the `listIR` template in succession for each IR instance contained in the PLITE instance. The `listIR` template either outputs all the VRML objects in the IR instance or one specified media object depending on the presence of the `arSelect` parameter. This way of modular design of the style sheet keeps the repetition of code to minimum.

In the current version of ARIFLite, browser caching of VRML objects for augmented reality is not done. It is important to note that this can be entirely controlled by the augmented reality templates without any modification in the source code of the ARIFLite application. Indicated in section 3.3.2.2, the augmented reality templates can be made to add the `cache` parameter at the end of each URL for an AR_OBJECT or AR_SUBOBJECT. The current template uses the following code to dynamically generate those URL entries.

```
<xsl:element name="URL">http://<xsl:value-of select="$host"/>:<xsl:value-of
select="$port"/><xsl:value-of
select="$servletPath"/>?modeID=2&amp;pID=<xsl:value-of
select="$pID"/>&amp;sID=<xsl:value-of select="$sID"/>&amp;irfID=<xsl:value-of
select="$irfID"/>&amp;filename=<xsl:value-of
select="DATA_SUB_MEDIA_OBJECT/FILE/FILE_LOCATION"/></xsl:element>
```

The following code will make the ARIFLite client to cache data. The downloaded data will be cached in the Internet Explorer cache on the client machine depending on its cache settings though.

```
<xsl:element name="URL">http://<xsl:value-of select="$host"/>:<xsl:value-of
select="$port"/><xsl:value-of
select="$servletPath"/>?modeID=2&amp;pID=<xsl:value-of
select="$pID"/>&amp;sID=<xsl:value-of select="$sID"/>&amp;irfID=<xsl:value-of
select="$irfID"/>&amp;cache=true&amp;filename=<xsl:value-of
select="DATA_SUB_MEDIA_OBJECT/FILE/FILE_LOCATION"/></xsl:element>
```

3.3.4 Deployment

I deployed AXTE on Apache Tomcat 4.1.29 for the initial tests. I also enabled extensive logging to facilitate easy debugging. I tested this deployment over client-server architecture on a single machine as well as networked computers. At a later stage, I also re-deployed it for test purposes on a stable version of Apache Tomcat 5. For the final tests, I used four presentations, two of which were governed by one style sheet and the other two by another style sheet. I also tested it on the ARIFLite communication code. The final presentation templates had two themes: Roman Palace and Anne of Cleves House with presentation data using Roman Palace exhibitions and Anne of Cleves House exhibitions exported from the ARCO database as XDE instances.

Some of the deployment websites within the Centre for VLSI and Computer Graphics intranet are listed below.

- http://139.184.100.45:8080/ (my laptop)
- http://139.184.101.52:8080/deploy/ (ARCOLite deployment server)

- http://139.184.100.210:8080/arcolite/ (ARCO laptop)

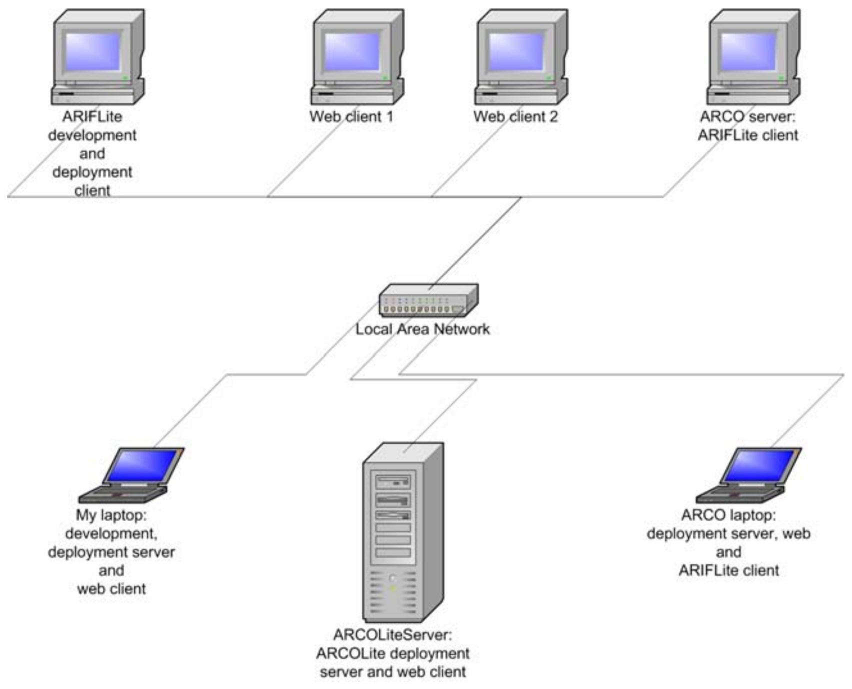

ARIFLite development and deployment client
Web client 1
Web client 2
ARCO server: ARIFLite client
Local Area Network
My laptop: development, deployment server and web client
ARCOLiteServer: ARCOLite deployment server and web client
ARCO laptop: deployment server, web and ARIFLite client

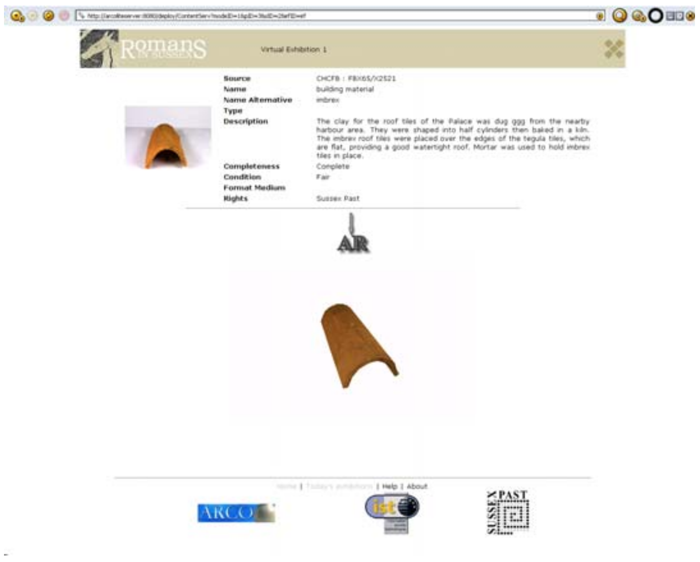
Figure 18. Approximate deployment diagram for ARCOLite test deployment without the content creation and content management sub-systems

Figure 18 shows the approximate test deployment scenario for ARCOLite that I used. This shows three computers on which ARCOLite server has been deployed, where as any computer in the deployment network can access any of these servers as clients. Some clients have ARIFLite application deployed on them as well.

3.4 Content visualisation: communication between exhibition server and ARIFLite

3.4.1 Overview

The content visualisation sub-system consists of a variety of web clients capable of rendering web content (XHTML and embedded media content). The XHTML content generated from the exhibition server adheres to the W3C XHTML specifications. Therefore, any standards compliant web browser capable of displaying graphics and embedded media (VRML for example) is able to display virtual museum exhibitions. Figure 19 shows the web content generated from the exhibition server and displayed in a standards compliant web browser like Mozilla 1.6 (http://www.mozilla.org/).



Source	CHCFB : FBX65/X2521
Name	building material
Name Alternative	imbrex
Type	
Description	The clay for the roof tiles of the Palace was dug ggg from the nearby harbour area. They were shaped into half cylinders then baked in a kiln. The imbrex roof tiles were placed over the edges of the tegula tiles, which are flat, providing a good watertight roof. Mortar was used to hold imbrex tiles in place.
Completeness	Complete
Condition	Fair
Format Medium	
Rights	Sussex Past

Figure 19. Web view in a standards compliant web browser (Mozilla)

To extend the functionality of such standards compliant web browsers to a higher level of interactivity on an augmented reality table-top, an integrated application called Augmented Reality Interface Lite (ARIFLite) has been developed. ARIFLite is a Windows GUI application based on Microsoft Foundation Classes as a wrapper for the Windows API. It also uses specialised libraries like AR Toolkit, OpenVRML, Microsoft DirectShow, etc. for the augmented reality system. ARIFLite application is a SDI (single document interface) with a split pane view. One of these panes encapsulate an embedded Internet Explorer browser (`CHtmlView` in MFC) while the other view is a sub-classed object of `CView` that uses memory buffer swapping functions to draw the contents of the augmented reality table-top. The contents of this view consist of a digital video camera buffer obtained through Microsoft DirectShow and the virtual reality objects drawn, using OpenGL, on the markers after they are detected through the AR Toolkit pattern recognition code.

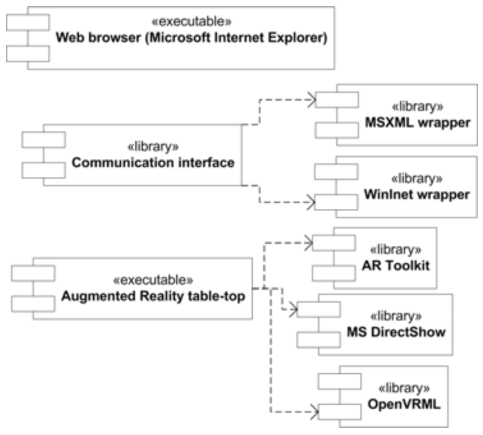


Figure 20. Components of ARIFLite

Figure 20 illustrates the three main components and their sub-components in the ARIFLite application.

- *Web browser* is one of the main end-user views in ARIFLite application. It is a sub-class of the built-in Microsoft Internet Explorer library (`CHtmlView` in MFC).

- *Communication interface* is a library, having further sub-components that provide a connectivity interface between the web view and the augmented reality table-top view. Illustrated in Figure 19, there are special links (the “AR” button) on the web content generated by the exhibition server that allow ARIFLite to obtain the necessary virtual reality models from the exhibition server. Once the user switches to the augmented reality view, these objects are placed in the augmented reality scene. This is when the communication code between the ARIFLite application and the exhibition server comes into play. This communication interface is a part of my final year project.
- *Augmented Reality table-top* is a complex interactive user view where a byte stream from a digital video camera is rendered on to MFC `CView` sub-class through Microsoft DirectShow libraries. This is a real-time video stream but lags may be visible depending on the camera capabilities and interface (USB 1.1 or IEEE1394). A pattern recognition system in the AR Toolkit library is used to detect markers in the video scene. On detection of relevant markers, objects (VRML data, text, images, etc.) can be superimposed on the video using OpenGL. This concept of interactive augmented reality is extensively discussed in [4].

For test purposes, there is a menu function called *Interchange Views* triggered by `SHIFT + I` from the keyboard that enables switching between the web view and augmented reality view; when one view is active, the other is hidden. This will be automated at a later stage. At a later stage of development of ARIFLite, a learning scenario view will also be introduced, which will display both the augmented reality view and the browser view at the same time to provide interactive learning context.

3.4.2 Architecture

The architecture of the communication interface between the augmented reality table-top and the web browser in ARIFLite can be summarised from Figure 20 in Figure 21.

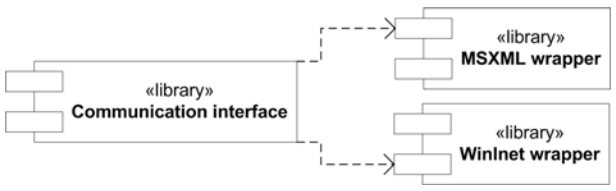


Figure 21. Communication interface between AR and web in ARIFLite

The communication interface encapsulates a specialised MSXML wrapper class and a MFC-based `WinInet` wrapper. I wrote the MSXML wrapper to integrate basic XML I/O using W3C DOM2 and basic XML transformation capabilities using XPath and XSLT engine.

The functionality of the communication interface in ARIFLite is illustrated by a UML state chart diagram in Figure 22. Since the augmented reality view is triggered from the web view, it is necessary in some way to make the web view understand that certain links mean that they are to be processed by the augmented reality connectivity code. I used a pseudo-protocol over HTTP to achieve this. I call this protocol as ARIF. This means that a URL like `arif://server:port/dir/file` will indicate to the web view that this URL needs to be handled by the AR connectivity code. The URLs of ARIF protocol point to the AXTE servlet on the exhibition server with parameters sufficient enough to generate instances of the ARDATA schema (see section 3.1).

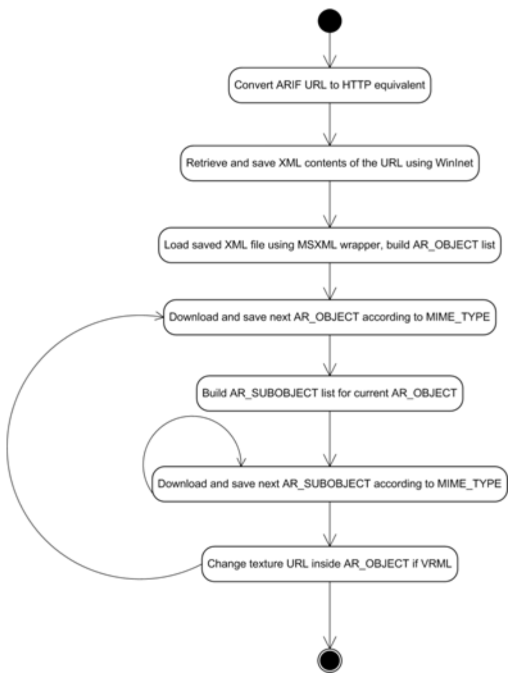

Convert ARIF URL to HTTP equivalent
Retrieve and save XML contents of the URL using WinInet
Load saved XML file using MSXML wrapper, build AR_OBJECT list
Download and save next AR_OBJECT according to MIME_TYPE
Build AR_SUBOBJECT list for current AR_OBJECT
Download and save next AR_SUBOBJECT according to MIME_TYPE
Change texture URL inside AR_OBJECT if VRML

Figure 22. State chart for the communication interface in ARIFLite

To begin with, the connectivity code in ARIFLite replaces `arif://` in the URL by `http://` equivalent. It then uses `WinInet` library to retrieve the hypertext content pointed by the URL through default proxy and cache settings of the Internet Explorer browser on the client machine. This content is an instance of the ARDATA schema. This XML data is saved as `ardata.xml` and loaded into a W3C DOM2 tree structure using the MSXML wrapper class library. A list of AR_OBJECT entries is built using XPath. Iterating through this list, each AR_OBJECT (such as a VRML file) is downloaded from the exhibition server pointed to by its URL entry. A list of AR_SUBOBJECT entries (if any) is built for the current AR_OBJECT. This list is iterated and each AR_SUBOBJECT is downloaded from the exhibition server. Once all the AR_SUBOBJECT files are downloaded for the current AR_OBJECT, the texture-URLs are updated if the current AR_OBJECT is a VRML file.

3.4.3 Implementation and results

The communication interface is implemented in two C++ classes (`CMSXML4Wrapper` and `CARIFLiteConnectitvity`). The following two code listings summarises the headers for these two classes respectively. Full source code listing is in section 9.2.1.

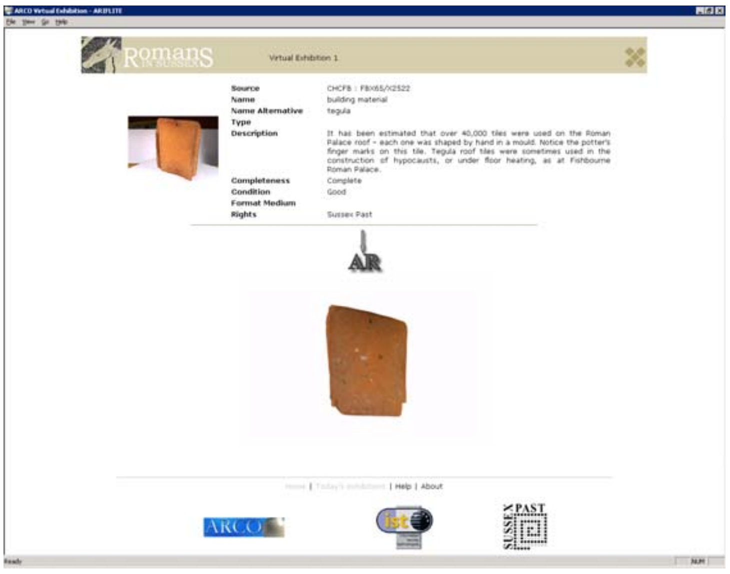
Class: CMSXML4Wrapper

```
#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#import <msxml4.dll>
using namespace MSXML2;

class CMSXML4Wrapper
{
private:
      IXMLDOMDocument2Ptr pXMLDom, pXSLStyle, pXMLOut;
      IXSLTemplatePtr pXSLTemplate;
      IXSLProcessorPtr pXSLTEngine;
      HRESULT hr;
      bool isLoaded;
public:
	CMSXML4Wrapper();
	virtual ~CMSXML4Wrapper();
	BOOL InitLibrary();
	BOOL LoadXMLFile(CString FilePath);
	void UnloadXML();
	BOOL SaveXMLFile(CString FilePath, IXMLDOMDocument2Ptr pDoc = NULL);
	IXMLDOMDocument2Ptr GetXMLDOMDoc();
	IXMLDOMNodePtr GetDOMNode(CString XPathExpression, IXMLDOMNodePtr
pStartHere = NULL);
	IXMLDOMNodeListPtr GetDOMNodes(CString XPathExpression, IXMLDOMNodePtr
pStartHere = NULL);

	BOOL PrepareXSLT(CString XSLStylesheetPath, IXMLDOMDocument2Ptr pDoc =
NULL);
	BOOL AddXSLParameters(CString paramName, CString paramValue);
	BOOL ApplyXSLT(CString XMLOutputPath);
};
```

Class: CARIFLiteConnectivity

```
#define ARPROTO "arif://"
#define HTTPPROTO "http://"
#define IO_CHUNKSIZE 65536 //optimise this number; for hard disks this can be
high but for slow networks?
#define ARIF_HTTP_USER_AGENT "Mozilla/4.0 (compatible; ARIFLite 1.0; Windows
(any); UoS, ARCO Consortium)"
#define ARIF_HTTP_VERSION "HTTP/1.1"
#define MAX_CACHE_ENTRY_INFO_SIZE 4096

//error codes
#define ARERR_SUCCESS 0 //no error
#define ARERR_NONET -1 //no internet connectivity
#define ARERR_NOHOSTCONNECT -2 //no connection to host
#define ARERR_NOHTTPREQUEST -3 //no HTTP request can be created
#define ARERR_NOHTTPREQUESTSENT -4 //HTTP request cannot be sent
#define ARERR_INVALIDURL -5 //invalid HTTP URL

#include <afxinet.h>

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CARIFLiteConnectivity : public CObject {
public:
	CARIFLiteConnectivity();
	BOOL RetrieveResource(char *lpszARUrl, const char *lpszLocalFilePath);
	int GetError();
	char *InterpretError(int nError);
private:
	int nErrorCode;
	CString vrmlData;
	int seekLocation;
	BOOL HandleARUrl(char *lpszARUrl);
	BOOL DownloadResource(char *lpszUrl, const char *lpszDestination,DWORD
flags = INTERNET_FLAG_TRANSFER_BINARY);
	void ReadVrmlFile(const char *lpszLocation);
	void WriteVrmlFile(const char *lpszLocation);
	void SetTextureUrl(const char *lpszReplace, const char *lpszUrl);
	void CleanupDirectory(const char *lpszSearchDir);
};
```

The communication code of the ARIFLite application has been tested over the example deployment of the exhibition server. Figure 23 shows ARIFLite in the browser mode. In this screenshot, a 3D virtual object from one example exhibition is displayed with the Fishbourne Roman Palace presentation style sheet.



Source	CHCFB : FBX65/X2522
Name	building material
Name Alternative	tegula
Type	
Description	It has been estimated that over 40,000 tiles were used on the Roman Palace roof - each one was shaped by hand in a mould. Notice the potter's finger marks on this tile. Tegula roof tiles were sometimes used in the construction of hypocausts, or under floor heating, as at Fishbourne Roman Palace.
Completeness	Complete
Condition	Good
Format Medium	
Rights	Sussex Past

Figure 23. ARIFLite in browser mode displaying a virtual museum artefact

On clicking the AR button, and then switching the ARIFLite application to augmented reality interface mode, the AR table top is displayed in Figure 24. It shows the same virtual museum object placed on a marker in the augmented reality scene.

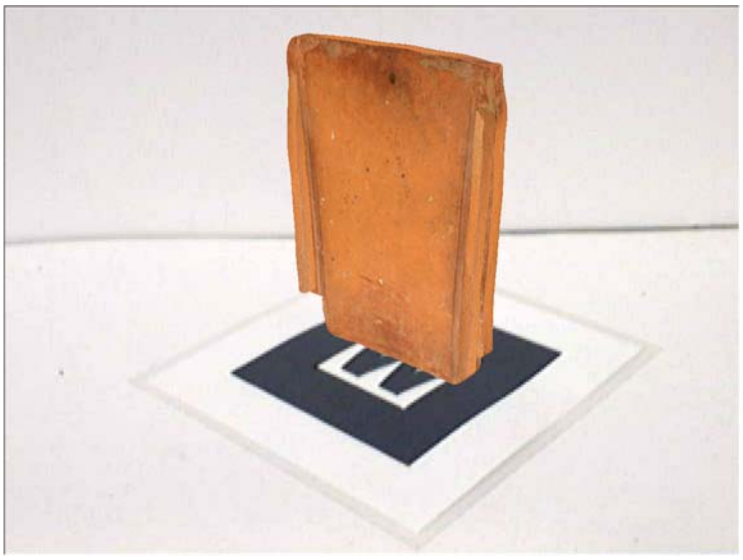


Figure 24. ARIFLite in full-screen augmented reality table-top mode displaying a virtual museum artefact on marker

4. Publications

As a part of this project, I have co-authored some research publications within the ARCO Consortium. These are listed below, in order of dates of publication or submission to appropriate authority.

- M White, F Liarokapis, N Mourkoussis, **A Basu**, J Darcy, P Petridis, and P Lister, A Lightweight XML driven architecture for the presentation of virtual cultural exhibitions (ARCOLite). *Appeared in IADIS Advanced Computing 2004 Conference*. Lisbon, Portugal 23-26 March, 2004.
- M White, F Liarokapis, N Mourkoussis, **A Basu**, J Darcy, P Petridis, M Sifniotis, and P Lister. ARCOLite – an XML based system for building and presenting Virtual Museum Exhibitions using Web3D and Augmented Reality. *To appear in EUROGRAPHICS 2004.* Bournemouth, UK. 8-10 June, 2004.
- M White, F Liarokapis, N Mourkoussis, **A Basu**, J Darcy, P Petridis, M Sifniotis, P Lister, Web3D and Augmented Reality support for Engineering Education. *Submitted to the journal World Transactions on Engineering Education*. 2004.
- M White, N Mourkoussis, F Liarokapis, **A Basu**, P Lister. ARCOLite — An architecture using XML, Web3D, Virtual and Augmented Reality to Present Digital Content. *Submitted to Computer Animation and Social Agents 2004 Conference.* MIRALab, University of Geneva. Geneva. Switzerland. 7-9 July, 2004.

5. Conclusions

With effective project planning, management and research, all the objectives stated in section 1.3 were achived by the project closing date March 12, 2004.

- The user behaviour analysis of the system were formally specified.
- The ARCO data model was analysed and tailored to the needs of the ARCOLite system. The ARCOLite XML repository was implemented.
- The exhibition server was implemented to work with the ARCOLite XML Repository.
- The communication interface between the exhibition server and the ARIFLite client was implemented.

In addition to the pre-determined objectives, the following tasks were done too.

- Extensions to the migrator were visualised, as stated in section 3.2.4.
- The project has been proved to be scalable over engineering education domains (publication referenced in [9]).
- I have co-authored four research publications (section 4), which have contributed a lot to my experience in a research environment.

By the end of this project, I have gained valueable experience and knowledge, which I am willing to apply in future developments in the Content Management and Content Visualisation sub-systems during the summer till the formal end date of the ARCO project in September 2004.

6. References

[1] Augmented Representation of Cultural Objects (ARCO) Consortium. http://www.arco-web.org/, 2001 – present.

[2] Walczak K, Cellary W. *Building Database Applications of Virtual Reality with X-VRML. Appeared in 7th International Conference on 3D Web Technology (Web3D 2002)*. Arizona, USA, February 2002.

[3] Wojciechowski R, Walczak K, White M. Augmented Reality Interface in Museum Artefact Visualization. *Appeared in IASTAD International Conference on Visualisation, Imaging and Image Processing VIIP 2003*. Benalmadena, Spain, September 2003.

[4] Kato H, Billinghurst M, et al. Virtual Object manipulation on table-top AR environment. *In proceedings of International Symposium on Augmented Reality, pp 111-199*. 2000.

[5] ARCO Consortium. D11 – Final report on XML Technology. http://www.arco-web.org/TextVersion/Documents/Deliverables/d11.html. September 2003.

[6] Network Working Group. RFC 2068 – Hypertext Transfer Protocol HTTP/1.1. http://www.ietf.org/rfc/rfc2068.txt. January 1997.

[7] Network Working Group. RFC 1521 – MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies. http://www.ietf.org/rfc/rfc1521.txt. September 1993.

[8] Network Working Group. RFC 1522 – MIME (Multipurpose Internet Mail Extensions) Part Two: Message Header Extensions for Non-ASCII Text. http://www.ietf.org/rfc/rfc1522.txt. September 1993.

[9] M White, F Liarokapis, N Mourkoussis, A Basu, J Darcy, P Petridis, M Sifniotis, P Lister, Web3D and Augmented Reality support for Engineering Education. *Submitted to the journal World Transactions on Engineering Education*. 2004.

[10] M White, F Liarokapis, N Mourkoussis, A Basu, J Darcy, P Petridis, and P Lister, A Lightweight XML driven architecture for the presentation of virtual cultural exhibitions (ARCOLite). *Appeared in IADIS Advanced Computing 2004 Conference*. Lisbon, Portugal 23-26 March, 2004.

[11] M White, F Liarokapis, N Mourkoussis, A Basu, J Darcy, P Petridis, M Sifniotis, and P Lister. ARCOLite – an XML based system for building and presenting Virtual Museum Exhibitions using Web3D and Augmented Reality. *To appear in EUROGRAPHICS 2004.* Bournemouth, UK. 8-10 June, 2004.

[12] N Mourkoussis, M White, M Patel, J Chmielewski, K Walczak. AMS – Metadata for Cultural Exhibitions using Virtual Reality. *Appeared in International Conference on Dublin Core and Metadata Applications 2003*. September – October 2003. Seattle, USA

[13] M White, N Mourkoussis, F Liarokapis, A Basu, P Lister. ARCOLite — An architecture using XML, Web3D, Virtual and Augmented Reality to Present Digital Content. *Submitted to Computer Animation and Social Agents 2004 Conference.* MIRALab, University of Geneva. Geneva. Switzerland. 7-9 July, 2004.

7. Appendix A – Project Proposal and Interim Report

7.1 Project proposal

7.1.1 Introduction

ARCO – Augmented Representation of Cultural Objects is a EU IST Framework V funded research project aimed at providing museums with useful technologies for digitising, managing and presenting virtual museum artefacts in virtual cultural environments. ARCOLite is a lightweight derivative of ARCO that eliminates both the database and the X-VRML server and uses standard XML technologies for its data repository and for content visualisation. This approach reduces the total cost of ownership making ARCOLite easily affordable by small and medium scale museums. In addition, ARCOLite demostrates a scalable interoperability prototype with any external digital culture system as well as learning scenario system. This final year project (*A visualisation system for viewing museum artefacts*) focuses on certain areas of the ARCOLite system.

7.1.2 Objectives

Having identified the scope of my final year project in the context of the ARCOLite system overview, I enlist the objectives of my final year project below.

- Specify the user behaviour analysis of the system.
- Conceptualise the data model and implement the ARCOLite XML data repository.
- Design and implement an interoperability tool – the migrator – for generating data for the ARCOLite XML data repository from an external system like ARCO.
- Design and implement the exhibition server. Specify the working principle behind the web and augmented reality presentation templates. Deploy the exhibition server with a sample presentation data and a sample presentation template.
- If time permits, then design and implement a communication interface between ARIFLite and the exhibition server.

7.1.3 Background work

I have been working in the ARCO research project as a summer internship during the July-September period of the year 2003. I have worked on areas of XML based content management system under development within the ARCO project. Before starting my final year project, I had done sufficient background work on XML technologies including design of schemas, XSL style sheets, etc.

Intense Java and C++ programming are required in my final year project. I had basic concepts of Java servlets before starting the project. I also have worked on C and C++ programming on Windows platform (mostly using MFC) in the past for my own pleasure, so I just had to refresh my memories before undertaking the parts of this project that required C++ programming.

7.1.4 Project plan

My project plan is shown in the following table.

Task ID	Task description	Date range
1	Evaluate project requirements and conceptualise the system-user interaction	October 2003 – November 2003
2	Visualise the data model. Implement the XML data repository. Design and implement the migrator tool.	November 2003 – December 2003 (before Christmas break)
3	Design and implement the exhibition server. Deploy the exhibition server with sample data. Test system functionality.	January 2004 – February 2004
4	Connect the exhibition server to the ARIFLite interface using a communication interface.	February 2004 – March 12, 2004 (end of project date)

7.2 Interim report

7.2.1 Work progress

At this stage of the project, the project requirements have been evaluated. I have implemented the XML data repository. I have implemented the migrator tool to obtain data for the XML data repository from the ARCO system. I have designed and implemented the exhibition server. At this point, there is a need for multiple presentations with multiple interoperable style sheets. Therefore, I need to make small changes in the exhibition server and amend the XML data repository to meet this requirement.

7.2.2 Amendments to objectives

The amendments to the previously stated objectives of the project are enlisted below.

- Specify the user behaviour analysis of the system.
- Update the data model (if necessary) and re-implement the ARCOLite XML data repository to meet the requirement of multiple presentations and multiple style sheets.
- Re-design and re-implement the exhibition server to meet the requirement of multiple presentations and multiple style sheets. Adapt the web presentation templates according to this need. Re-deploy the exhibition server with sample presentation data and sample presentation templates.
- Design and implement a communication interface between ARIFLite and the exhibition server.

7.2.3 Updated project plan

Task ID	Task description	Date range
1	Evaluate project requirements and conceptualise the system-user interaction	October 2003 – November 2003
2	Visualise the data model. Implement the XML data repository. Design and implement the migrator tool.	November 2003 – December 2003 (before Christmas break)
3	Design and implement the exhibition server. Deploy the exhibition server with sample data. Test system	December 2003 (till the

	functionality.	Christmas break)
4	Re-implement the XML data repository and re-deploy the exhibition server with amendments to the servlet architecture to meet the requirements of multiple presentations with multiple interoperable style sheets.	January 2004 – February 2004
5	Connect the exhibition server to the ARIFLite interface using a communication interface.	February 2004 – March 12, 2004 (end of project date)

8. Appendix B – XML schemas

8.1 XSD for IR

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
	<xs:element name="INFORMATION_RESOURCE">
		<xs:complexType>
			<xs:sequence>
				<xs:element name="DATA">
					<xs:complexType>
						<xs:sequence>
							<xs:element name="THUMBNAIL"
minOccurs="0">
								<xs:complexType>
									<xs:sequence>
										<xs:element
ref="FILE_LOCATION"/>
									</xs:sequence>
								</xs:complexType>
							</xs:element>
							<xs:element name="METADATA">
								<xs:complexType>
									<xs:sequence>
										<xs:element
ref="GRAMMAR"/>
										<xs:element
ref="CODE"/>
									</xs:sequence>
								</xs:complexType>
							</xs:element>
							<xs:element ref="MEDIA_OBJECT"
minOccurs="0" maxOccurs="unbounded"/>
						</xs:sequence>
					</xs:complexType>
				</xs:element>
				<xs:element ref="ENTRY_ID"/>
				<xs:element name="GRAMMAR_TYPE">
					<xs:simpleType>
						<xs:restriction base="xs:string">
							<xs:enumeration value="Schema"/>
							<xs:enumeration value="DTD"/>
						</xs:restriction>
					</xs:simpleType>
				</xs:element>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="FILE_LOCATION" type="xs:string"/>
	<xs:element name="GRAMMAR">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="FILE_LOCATION"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="CODE">
		<xs:complexType>
			<xs:sequence>
				<xs:element ref="FILE_LOCATION"/>
			</xs:sequence>
		</xs:complexType>
	</xs:element>
	<xs:element name="FILE">
		<xs:complexType>
			<xs:sequence>
```

```
                          <xs:element ref="FILE_LOCATION"/>
                          <xs:element ref="MIME_TYPE" minOccurs="0"/>
                    </xs:sequence>
              </xs:complexType>
       </xs:element>
       <xs:element name="SUB_MEDIA_OBJECT">
              <xs:complexType>
                    <xs:sequence>
                          <xs:element name="DATA_SUB_MEDIA_OBJECT">
                                 <xs:complexType>
                                       <xs:sequence>
                                              <xs:element ref="FILE"/>
                                              <xs:element name="METADATA"
minOccurs="0">
                                                    <xs:complexType>
                                                          <xs:sequence>
                                                                 <xs:element
ref="GRAMMAR" minOccurs="0"/>
                                                                 <xs:element
ref="CODE"/>
                                                          </xs:sequence>
                                                    </xs:complexType>
                                              </xs:element>
                                       </xs:sequence>
                                 </xs:complexType>
                          </xs:element>
                          <xs:element ref="ENTRY_ID"/>
                    </xs:sequence>
              </xs:complexType>
       </xs:element>
       <xs:element name="MEDIA_OBJECT">
              <xs:complexType>
                    <xs:sequence>
                          <xs:element name="DATA_MEDIA_OBJECT">
                                 <xs:complexType>
                                       <xs:sequence>
                                              <xs:element ref="FILE"
minOccurs="0"/>
                                              <xs:element name="METADATA"
minOccurs="0">
                                                    <xs:complexType>
                                                          <xs:sequence>
                                                                 <xs:element
ref="GRAMMAR" minOccurs="0"/>
                                                                 <xs:element
ref="CODE"/>
                                                          </xs:sequence>
                                                    </xs:complexType>
                                              </xs:element>
                                              <xs:element
ref="SUB_MEDIA_OBJECT" minOccurs="0" maxOccurs="unbounded"/>
                                       </xs:sequence>
                                 </xs:complexType>
                          </xs:element>
                          <xs:element ref="ENTRY_ID"/>
                    </xs:sequence>
              </xs:complexType>
       </xs:element>
       <xs:element name="ENTRY_ID" type="xs:string"/>
       <xs:element name="MIME_TYPE" type="xs:string"/>
</xs:schema>
```

8.2 XSD for PLITE

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
       <xs:element name="EXHIBITION">
              <xs:complexType>
                    <xs:sequence>
```

```
                          <xs:element name="DATA" maxOccurs="unbounded">
                                <xs:complexType>
                                      <xs:sequence>
                                            <xs:element
ref="FILE_LOCATION"/>
                                      </xs:sequence>
                                </xs:complexType>
                          </xs:element>
                          <xs:element name="ENTRY_ID" type="xs:string"/>
                          <xs:element name="NAME" type="xs:string"/>
                          <xs:element name="DESCRIPTION" type="xs:string"/>
                          <xs:element name="BANNER" minOccurs="0">
                                <xs:complexType>
                                      <xs:sequence>
                                            <xs:element
ref="FILE_LOCATION"/>
                                      </xs:sequence>
                                </xs:complexType>
                          </xs:element>
                    </xs:sequence>
              </xs:complexType>
      </xs:element>
      <xs:element name="FILE_LOCATION" type="xs:string"/>
</xs:schema>
```

8.3 XSD for IRF

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
      <xs:element name="IRF">
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="PRESENTATION"
maxOccurs="unbounded">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element name="ID"
type="xs:int"/>
                                          <xs:element name="MAIN_DATA"
type="xs:string"/>
                                          <xs:element name="DATA_PATH"
type="xs:string"/>
                                    </xs:sequence>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="STYLESHEET" maxOccurs="unbounded">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element name="ID"
type="xs:int"/>
                                          <xs:element
name="STYLESHEET_PATH" type="xs:string"/>
                                          <xs:element name="MAIN_STYLE"
type="xs:string"/>
                                    </xs:sequence>
                              </xs:complexType>
                        </xs:element>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

8.4 XSD for ARDATA

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
      <xs:element name="ARDATA">
            <xs:complexType>
```

```
                    <xs:sequence>
                          <xs:element name="AR_OBJECT" maxOccurs="unbounded">
                                <xs:complexType>
                                      <xs:sequence>
                                            <xs:element ref="URL"/>
                                            <xs:element ref="MIME_TYPE"/>
                                            <xs:element name="AR_SUBOBJECT"
minOccurs="0" maxOccurs="unbounded">
                                                  <xs:complexType>
                                                        <xs:sequence>
                                                              <xs:element
ref="URL"/>
                                                              <xs:element
ref="MIME_TYPE"/>
                                                        </xs:sequence>
                                                  </xs:complexType>
                                            </xs:element>
                                      </xs:sequence>
                                </xs:complexType>
                          </xs:element>
                    </xs:sequence>
              </xs:complexType>
      </xs:element>
      <xs:element name="URL" type="xs:string"/>
      <xs:element name="MIME_TYPE" type="xs:string"/>
</xs:schema>
```

9. Appendix C – source code

9.1 Java code

9.1.1 Migrator

9.1.1.1 ARCOLiteException.java

```
package migration;


/**
 * <p>Title: ARCO Platform Migration Tool</p>
 * <p>Description: Tool for migrating between ARCO platforms.</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: University of Sussex, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0 (Build: November 2003)
 */

public class ARCOLiteException
    extends Throwable {
  private Throwable theCause;
  protected ARCOLiteException(Throwable cause) {
    /*
      Maintains an internal Throwable object
     instead of normal subclassing of the Throwable
     class.
     */
    theCause = cause;
  }

  /**
   * Overrides the superclass method and redefines
   * it in the context of this exception class.
   *
   * @return returns the underlying Throwable object
   * that caused the exception.
   */
  public Throwable getCause() {
    return theCause;
  }

  /**
   * Similar to its superclass method, this method
   * prints the stack trace for the underlying Throwable
   * object into System.out.
   */
  public void printStackTrace() {
    theCause.printStackTrace();
  }

  /**
   * Similar to its superclass method, this method
   * prints the stack trace for the underlying Throwable
   * object through the specified PrintWriter object.
   *
   * @param writer java.io.PrintWriter object through which
   * the stack trace is printed.
   */
  public void printStackTrace(java.io.PrintWriter writer) {
    theCause.printStackTrace(writer);
  }

  /**
```

```
   * Similar to its superclass method, this method
   * prints the stack trace for the underlying Throwable
   * object into the specified print stream.
   *
   * @param stream java.io.PrintStream object into which
   * the stack trace is serialized.
   */
  public void printStackTrace(java.io.PrintStream stream) {
    theCause.printStackTrace(stream);
  }

  /**
   * Obtains the stack trace as an array of StackTraceElement
   * objects from the underlying Throwable object.
   *
   * @return array of StackTraceElement objects containing
   * all the information regarding the stack trace.
   */
  public StackTraceElement[] getStackTrace() {
    return theCause.getStackTrace();
  }

  /**
   * Similar to all exception classes, this method obtains
   * the descriptive message from the underlying Throwable
   * object.
   *
   * @return String object containing the descriptive message
   * for the exception.
   */
  public String getMessage() {
    return theCause.getMessage();
  }
}
```

9.1.1.2 ARCOLiteSAXErrorHandler.java

```
package migration;

/**
 * <p>Title: ARCO Platform Migration Tool</p>
 * <p>Description: Tool for migrating between ARCO platforms.</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: University of Sussex, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0 (Build: November 2003)
 */

import java.io.*;

import org.xml.sax.*;

/**
 * <P>
 * <STRONG>ARCOLiteSAXErrorHandler</STRONG> implements the
 * org.xml.sax.ErrorHandler interface for use with the built
 * in DOM parser of this package. This class uses the error
 * handling rules as specified in W3C XML 1.0 specifications.
 * Messages are printed out to output stream. At a later stage, RMI
 * based remote server logging may be introduced.
 * </P>
 */
class ARCOLiteSAXErrorHandler
    implements ErrorHandler {
  private boolean strict = false;
  private PrintWriter oStream;
  public ARCOLiteSAXErrorHandler() {
    oStream = new PrintWriter(System.out);
  }
  public ARCOLiteSAXErrorHandler(Writer output) {
    oStream = new PrintWriter(output);
  }
  /**
   * According to W3C XML 1.0 specifications, this method prints
   * out the information for a non-fatal error to the output stream and
   * lets the parser continue. If the error handler is in less
   * strict mode, this message is not printed to output stream.
   *
   * @param exception the SAXParseException object that is reported
   * by the DOM parser.
   */
  public void error(SAXParseException exception) {
    if (!strict) {
      return; //less-strict mode
    }
    oStream.println("[SAX error]: " + exception.getMessage());
    oStream.flush();
  }
  /**
   * According to the W3C XML 1.0 specifications, this method should
   * stop the parser from parsing. It does so by throwing an exception
   * and not printing out a fatal error message to the output stream.
   *
   * @param exception the SAXParseException object that is reported
   * by the DOM parser
   * @throws SAXException the SAXException thrown to stop the parser
   * from parsing.
   */
  public void fatalError(SAXParseException exception)
      throws SAXException {
    oStream.println("[SAX fatal-error]: " + exception.getMessage());
    oStream.println("Parse process halted");
    oStream.flush();
    throw new SAXException(exception);
  }
  /**
   * According to the W3C XML 1.0 specifications, this method prints out
   * the information for a parser warning to the output stream and lets the
   * parser continue. If the error handler is in less
   * strict mode, this message is not printed to output stream.
   *
   * @param exception the SAXParseException object that is reported by
   * the DOM parser.
   */
  public void warning(SAXParseException exception) {
    if (!strict) {
      return; //less-strict mode
    }
    oStream.println("[SAX warning]: " + exception.getMessage());
    oStream.flush();
  }
  /**
   * This method is used to toggle the strictness of error
   * handling. By default the error handler is in less
   * strict mode and only reports fatal errors when parsing
   * is aborted.
   *
   * @param flag a boolean flag to indicate if the error handler
   * should be strict.
   */
  public void makeStrict(boolean flag) {
    strict = flag;
  }
}
```

9.1.1.3 ARCOLiteXSLTErrorListener.java

```
package migration;


/**
 * <p>Title: ARCO Platform Migration Tool</p>
 * <p>Description: Tool for migrating between ARCO platforms.</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: University of Sussex, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0 (Build: November 2003)
 */

import javax.xml.transform.*;


class ARCOLiteXSLTErrorListener
    implements ErrorListener {
  public void warning(TransformerException exception)
      throws TransformerException {
    //shut up and don't show messages :-P
  }

  public void error(TransformerException exception)
      throws TransformerException {
    System.out.println("[XSLT Error]: " + exception.getLocationAsString());
    System.out.flush();
  }

  public void fatalError(TransformerException exception)
      throws TransformerException {
    System.out.println("[XSLT Fatal Error]: " +
                    exception.getLocationAsString());
    System.out.flush();
    System.exit(-1);
  }
}
```

9.1.1.4 BatchProcess.java (the main migrator executable class)

```
package migration;


/**
 * <p>Title: ARCO Platform Migration Tool</p>
 * <p>Description: Tool for migrating between ARCO platforms.</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: University of Sussex, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0 (Build: November 2003)
 */

import java.io.*;
import java.util.*;


public class BatchProcess {
  public static void main(String[] args) {
    Vector coIDs;
    long startTime;
    try {
      if (args.length < 3) {
       System.out.println("Usage:\nbatch <xde file> <migration template>" +
                        " <plitetemplate>");
       System.exit( -1);
      }
      startTime = System.currentTimeMillis();
      System.out.println("\nARCOLite XDE to XDELite migration process
started");
      FileSystemFunctions fsObj = new FileSystemFunctions();
      fsObj.renameFSObject("arco_data","ARCO_DATA"); //rename the directory
      XDEProcessor xdeP = new XDEProcessor();
      coIDs = xdeP.getCOIDs(args[0]);
      if (coIDs.size() <= 0) {
       System.out.println("No cultural object found!");
       System.exit( -1);
      }
      Hashtable paramMap = new Hashtable();
      for (int i = 0; i < coIDs.size(); i++) {
       System.out.println();
       System.out.println("Processing cultural object " + (i+1) + " of " +
                        coIDs.size() + " cultural objects...");
       System.out.println("Selecting cultural object ID = " + coIDs.get(i));
       paramMap.put("coSelect", coIDs.get(i));
       System.out.flush();
       FileReader fXML = new FileReader(args[0]); //this is the XML file
       FileReader fXSL = new FileReader(args[1]); //this is the XSL file
       String outPath = "IR-" + coIDs.get(i) + "-XDELITE";
       FileWriter fXMLo = new FileWriter(outPath +
                                    ".XML"); //this is the XML file (out)
       XSLTEngine xslt = new XSLTEngine();
       System.out.println("Transforming XML...");
       System.out.flush();
       xslt.transformXSL(fXML, fXSL, fXMLo, paramMap);
      }
      System.out.println("\nGenerating Presentation LITE ...");
      System.out.flush();
      String pliteConvArgs[] = new String[7];
      pliteConvArgs[0] = args[0];
      pliteConvArgs[1] = args[2];
      pliteConvArgs[2] = "plite.xml";
      pliteConvArgs[3] = "file_pre";
      pliteConvArgs[4] = "IR-";
      pliteConvArgs[5] = "file_suf";
      pliteConvArgs[6] = "-XDELITE.XML";
      xmlconv.showLog = true; //for less info on the console
      xmlconv.doCleanup = false; //for fast processing don't call gc
      xmlconv.main(pliteConvArgs);
      System.out.println("\nCleaning up...");
      paramMap = null;
      coIDs = null;
      System.gc();
      System.out.println("ARCOLite migration finished in " +
                       (System.currentTimeMillis() - startTime) +
                       " milliseconds.");
      System.out.flush();
    }
    catch (Throwable th) {
      th.printStackTrace(System.out);
    }
  }
}
```

9.1.1.5 FileSystemFunctions.java

```
package migration;


/**
 * <p>Title: ARCO Platform Migration Tool</p>
 * <p>Description: Tool for migrating between ARCO platforms.</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: University of Sussex, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0 (Build: November 2003)
 */

import java.io.*;
import java.text.*;
import java.util.*;


class FileSystemFunctions {
  /**
   * <P>
   * This method creates a new directory in the path
   * specified.
   * </P>
   * @param dirPath String object specifying absolute path
   * for the new directory or the directory name in the current
   * path.
   * @return boolean value indicating if the operation is
   * successful.
   * @throws ARCOLiteException a generalized exception for
   * the package thrown when fatal errors are encountered while
   * working with the file system objects.
   */
  public boolean makeNewDirectory(String dirPath)
      throws ARCOLiteException {
    try {
      File directory = new File(dirPath);
      /*if(directory.isDirectory() == false) {
       throw new ARCOLiteException(new Throwable(
           dirPath + " is not a directory."));
            }
            If the directory is to be created, how do you check if it is a
            directory?????!!!!!return
      */
     directory.mkdir();
     return true;
    }
    catch (Throwable e) {
      throw new ARCOLiteException(e);
    }
  }

  /**
   * <P>
   * This method renames a file or a directory (file system object).
   * </P>
   * @param oldPath String object specifying the absolute path or
   * object name in the current path of the file system object to
   * be renamed.
   * @param newPath String object specifying the absolute path or
   * the object name in the current path of the file system object
   * to which the existing file system object is to be renamed.
   * @return boolean value to indicate if the operation is successful.
   * @throws ARCOLiteException a generalized exception for
   * the package thrown when fatal errors are encountered while
   * working with the file system objects.
   */
  public boolean renameFSObject(String oldPath, String newPath)
      throws ARCOLiteException {
    try {
      File oldFSO = new File(oldPath);
      if (oldFSO.exists() == false) {
       throw new ARCOLiteException(
                new Throwable("File not found - " + oldPath));
      }
      return oldFSO.renameTo(new File(newPath));
    }
    catch (Throwable e) {
      throw new ARCOLiteException(e);
    }
  }

  /**
   * <P>
   * This method deletes a file or directory specified by the path.
   * </P>
   * @param pathToDelete String object specifying the absolute path
   * or object name in the current path of the file system object to
   * be deleted.
   * @return boolean value to indicate if the operation if successful.
   * @throws ARCOLiteException a generalized exception for
   * the package thrown when fatal errors are encountered while
   * working with the file system objects.
   */
  public boolean deleteFSObject(String pathToDelete)
```

```
      throws ARCOLiteException {
    try {
      File deleteThis = new File(pathToDelete);
      if (deleteThis.exists() == false) {
       throw new ARCOLiteException(new Throwable("File not found - " +
                                   pathToDelete));
      }
      if (deleteThis.canWrite() == false) {
       throw new ARCOLiteException(new Throwable("Write access denied on " +
                                   pathToDelete));
      }
      return deleteThis.delete();
    }
    catch (Throwable e) {
      throw new ARCOLiteException(e);
    }
  }

  /**
   * <P>
   * This method copies a file from one location to another. This
   * operation uses byte I/O. No character translation is done.
   * </P>
   * @param oldLocation String object specifying the absolute path
   * or the file name in the current path of the file that is to be
   * copied.
   * @param newLocation String object specifying the absolute path
   * or the file name in the current path of the file to which the
   * file data is to be copied.
   * @return long value to indicate the time in milliseconds expended
   * by the virtual machine in copying the file.
   * @throws ARCOLiteException a generalized exception for
   * the package thrown when fatal errors are encountered while
   * working with the file system objects.
   */
  public long copyFile(String oldLocation, String newLocation)
      throws ARCOLiteException {
    try {
      File existing = new File(oldLocation);
      File newone = new File(newLocation);
      if (existing.exists() == false) {
       throw new ARCOLiteException(new Throwable("File not found - " +
                                   oldLocation));
      }
      if (existing.canRead() == false) {
       throw new ARCOLiteException(new Throwable("Read access denied on " +
                                   oldLocation));
      }
      if (newone.canWrite() == false) {
       throw new ARCOLiteException(new Throwable("Write access denied on " +
                                   newLocation));
      }
      FileInputStream fIn = new FileInputStream(existing);
      FileOutputStream fOut = new FileOutputStream(newone);
      long start = System.currentTimeMillis();
      for (; fIn.available() > 0; ) {
       //read and write one byte at a time
       fOut.write(fIn.read());
      }
      fIn.close();
      fOut.flush();
      fOut.close();
      return (System.currentTimeMillis() - start);
    }
    catch (Throwable e) {
      throw new ARCOLiteException(e);
    }
  }

  /**
   * <P>
```

```
    * This method obtains the size (in bytes) of the
    * requested file system object.
    * </P>
    * @param fileName relative or absolute path of the file
    * system object whose size is to be returned.
    * @return long value containing the size of the file system
    * object.
    * @throws ARCOLiteException a generalized exception for
    * the package thrown when fatal errors are encountered while
    * working with the file system objects.
    */
  public long getFileSize(String fileName)
      throws ARCOLiteException {
    try {
      File fsO = new File(fileName);
      if (fsO.exists() == false) {
       throw new ARCOLiteException(new Throwable("File not found - " +
                                           fileName));
      }
      if (fsO.isFile() == false) {
       throw new ARCOLiteException(new Throwable(fileName + " is not a
file."));
      }
      return fsO.length();
    }
    catch (Throwable e) {
      throw new ARCOLiteException(e);
    }
  }

  /**
   * <P>
   * This method obtains the last modified property of the
   * file in milliseconds counted from 00:00:00 GMT 01 JAN 1970.
   * </P>
   * @param fileName relative or absolute path of the file
   * system object.
   * @return long value the time in milliseconds to specify
   * the last modified property of the file.
   * @throws ARCOLiteException a generalized exception for
   * the package thrown when fatal errors are encountered while
   * working with the file system objects.
   */
  public long getLastModified(String fileName)
      throws ARCOLiteException {
    try {
      File fsO = new File(fileName);
      if (fsO.exists() == false) {
       throw new ARCOLiteException(new Throwable("File not found - " +
                                           fileName));
      }
      if (fsO.isFile() == false) {
       throw new ARCOLiteException(
                 new Throwable(fileName + " is not a file."));
      }
      return fsO.lastModified();
    }
    catch (Throwable e) {
      throw new ARCOLiteException(e);
    }
  }

  public String getTimeBasedFilePrefix() {
    Calendar now = Calendar.getInstance();
    SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd-hh-mm-ss-S");
    return format.format(now.getTime());
  }

  public String getDirPathFromFilePath(String filePath) {
    char delimC;
    //determine the path separator convention
    if(filePath.indexOf('/') > 0) {
      delimC = '/';
    }
    else {
      delimC = '\\';
    }
    int delim = filePath.lastIndexOf(delimC);
    if (delim == 0) {
      return null;
    }
    else {
      return filePath.substring(0,delim);
    }
  }

  public boolean isDirectory(String path) {
    File fTest = new File(path);
    return fTest.isDirectory();
  }

}
```

9.1.1.6 RawXMLReader.java

```
package migration;

/**
 * <p>Title: ARCO Platform Migration Tool</p>
 * <p>Description: Tool for migrating between ARCO platforms.</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: University of Sussex, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0 (Build: November 2003)
 */

import java.io.*;
import javax.xml.parsers.*;

import org.apache.xpath.*;

import org.w3c.dom.*;
import org.xml.sax.*;

/**
 * <P><CODE>RawXMLReader</CODE> is a class used by the
 * ARCO XML processor superstructure built to process the
 * XDE or metadata instances. This class lets the programmer
 * obtain the raw XML document using DOM. It is not to be
 * used by the GUI classes. This class is just used to isolate
 * the XML parsing code that is used by several other classes
 * in this package.</P>
 * <P>The complementary class of this class is <CODE>
 * RawXMLWriter</CODE>, which writes a DOM document into
 * an output stream or reader.
 * </P>
 * @author Anirban Basu (ab25@sussex.ac.uk)
 * @version 1.0.0.0
 */
class RawXMLReader {
  private DocumentBuilderFactory myFactory;
  private DocumentBuilder myBuilder;
  private Document myDOMDocument;
  private Node theRoot;
  private ARCOLiteSAXErrorHandler errorHandler;

  public RawXMLReader() {
    //No need to write Javadoc comments for this constructor
    myFactory = myFactory.newInstance();
    errorHandler = new ARCOLiteSAXErrorHandler();
  }
  /**
```

```
   * <P>
   * As the name suggests, this method is called before parsing
   * an XML document. The parser settings may be changed every time
   * before the document is parsed.
   * </P>
   *
   * @param expandEntityReferences boolean flag indicating
   * if entity references should be expanded and presented in
   * the DOM tree.
   * @param ignoreComments boolean flag indicating if comment
   * nodes should be ignored by the parser.
   * @param namespaceAware boolean flag indicating if the parser
   * should be namespace considerate.
   * @param validateWhileParsing boolean flag indicating if the
   * parser should validate the XML file. This is just the well-
   * formedness validation. It does not validate against the
   * schema.
   * @param strictErrorReporting boolean flag indicating if the
   * parser should report warnings and non-fatal errors.
   * @throws ARCOLiteException a generalized exception for
   * this package if there is any problem with pre-setting the
   * parser.
   */
  public void preParseInit(boolean expandEntityReferences,
                      boolean ignoreComments,
                      boolean namespaceAware,
                      boolean validateWhileParsing,
                      boolean strictErrorReporting)
      throws ARCOLiteException {
    try {
      myFactory.setExpandEntityReferences(expandEntityReferences);
      myFactory.setIgnoringComments(ignoreComments);
      myFactory.setNamespaceAware(namespaceAware);
      myFactory.setValidating(validateWhileParsing);
      myBuilder = myFactory.newDocumentBuilder();
      errorHandler.makeStrict(strictErrorReporting);
      myBuilder.setErrorHandler(errorHandler);
    }
    catch (Throwable e) {
      throw new ARCOLiteException(e);
    }
  }

  /**
   * <P>
   * This method parses an XML file based on the parser
   * configuration set by preParseInit method.
   * </P>
   *
   * @param xmlLocation an Object describing the location of
   * the XML file to be parsed. This can be an URI expressed
   * in the form of a String or a java.io.File object or a
   * java.io.InputStream object or org.xml.sax.InputSource
   * object.
   * @throws ARCOLiteException a generalized exception for
   * this package thrown if there is any problem with parsing.
   */
  public void parseAndPostProcess(Object xmlLocation)
      throws ARCOLiteException {
    /*
     The use of instanceof to determine the class of
     an object at runtime is similar to Runtime Type
     Identification (RTTI) in C++.
     */
    try {
      if (xmlLocation instanceof String) {
       //The location is an URI
       myDOMDocument = myBuilder.parse( (String) xmlLocation);
      }
      else if (xmlLocation instanceof File) {
       myDOMDocument = myBuilder.parse( (File) xmlLocation);
      }
      else if (xmlLocation instanceof InputSource) {
       myDOMDocument = myBuilder.parse( (InputSource) xmlLocation);
      }
      else if (xmlLocation instanceof InputStream) {
       myDOMDocument = myBuilder.parse( (InputStream) xmlLocation);
      }
      else {
       //Nothing else is allowed. Exception should be thrown.
       Throwable myExp = new Throwable("Unknown XML input source type.");
       throw new ARCOLiteException(myExp);
      }
      //Get the top-level element to form the DOM tree
      theRoot = (Node) myDOMDocument;
    }
    catch (Throwable e) {
      throw new ARCOLiteException(e);
    }
  }
  /**
   * <P>
   * This method returns the underlying DOM document
   * created through parsing an XML file.
   * </P>
   *
   * @return org.w3c.dom.Document object that contains
   * information about the parsed XML file.
   */
  public Document getDOMDocument() {
    return myDOMDocument;
  }
  /**
   * <P>
   * This method returns the top-level node (root) of
   * the DOM tree created through parsing an XML file.
   * </P>
   *
   * @return org.w3c.dom.Node object that is the root
   * of the DOM tree.
   */
  public Node getTheRootOfDOMTree() {
    return theRoot;
  }
  public Node findElement(String elementName,
                    Node startHere) {
    NodeList nList = startHere.getChildNodes();
    for(int i=0; i<nList.getLength(); i++) {
      if(nList.item(i).getNodeName().compareTo(elementName)==0 //match name
       && nList.item(i).getNodeType() == Node.ELEMENT_NODE) { //match type
        return nList.item(i);
      }
      else {
       Node depthSearch = findElement(elementName,nList.item(i));
       if(depthSearch!=null) return depthSearch;
      }
    }
    return null;
  }
  public Node findElementUsingXPath(String xpathExpression,
                               Node startHere)
      throws ARCOLiteException {
    try {
      return XPathAPI.selectSingleNode(startHere, xpathExpression);
    }
    catch (Throwable e) {
      throw new ARCOLiteException(e);
    }
  }
  protected String getValueOfElement(Node element) {
    NodeList nl = element.getChildNodes();
    for (int i = 0; i < nl.getLength(); i++) {
      if (nl.item(i).getNodeType() == Node.TEXT_NODE) {
       return nl.item(i).getNodeValue();
      }
    }
    return null;
  }
}
```

9.1.1.7 XDEProcessor.java

```
package migration;


/**
 * <p>Title: ARCO Platform Migration Tool</p>
 * <p>Description: Tool for migrating between ARCO platforms.</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: University of Sussex, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0 (Build: November 2003)
 */

import java.io.*;

import org.w3c.dom.*;
import java.util.*;


class XDEProcessor {
  private RawXMLReader xmlReader;
  private Vector theCOIDs;
  XDEProcessor()
      throws ARCOLiteException {
    xmlReader = new RawXMLReader();
    xmlReader.preParseInit(false, true, false, false, false);
    theCOIDs = new Vector();
  }

  private void loadXDE(String location)
      throws ARCOLiteException {
    File fXDE = new File(location);
    xmlReader.parseAndPostProcess(fXDE);
  }

  private Element getElement(String elementName)
      throws ARCOLiteException {
    Node findNode = xmlReader.findElement(elementName,
                                 xmlReader.getTheRootOfDOMTree());
    if (findNode != null) {
      return (Element) findNode;
    }
    else {
      return null;
    }
  }
  private Element getElementUsingXPath(String xpathExpression)
      throws ARCOLiteException {
    Node findNode = xmlReader.findElementUsingXPath(xpathExpression,
                                 xmlReader.getTheRootOfDOMTree());
    if (findNode != null) {
      return (Element) findNode;
    }
    else {
      return null;
    }
  }

  public Vector getCOIDs(String xdeLocation)
      throws ARCOLiteException {
    loadXDE(xdeLocation);
    Element firstCO = getElementUsingXPath(
       "//ARCO_DATA/DYNAMIC_DATA/CULTURAL_OBJECTS/CO");
    if(firstCO == null) {
      throw new ARCOLiteException(
         new Throwable("Cannot find batch process elements!"));
    }
    Node currentCO = firstCO;
    while(true) {
      if (currentCO == null) break;
      NamedNodeMap nMap = currentCO.getAttributes();
      if (nMap != null) {
       Node coID = nMap.getNamedItem("CO_ID");
       if (coID != null) {
         String coIDVal = coID.getNodeValue();
         theCOIDs.add(coIDVal);
       }
      }
      currentCO = currentCO.getNextSibling();
    }
    return theCOIDs;
  }
}
```

9.1.1.8 *xmlconv.java (independent command line XSL transformer)*

```
package migration;

/**
 * <p>Title: ARCO Platform Migration Tool</p>
 * <p>Description: Tool for migrating between ARCO platforms.</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: University of Sussex, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0 (Build: November 2003)
 */

import java.util.Hashtable;
import java.io.*;

public class xmlconv {
  public static boolean showLog = true;
  public static boolean doCleanup = true;
  public static void main(String[] args) {
    try {
      if (args.length < 3) {
       System.out.println("Usage:\n" +
           "xmlconv <xmlfile> <xslfile> <xmloutputfile> [[<key> <value>] " +
           "[<key> <value>]... ]");
       System.exit( -1);
      }
      Hashtable argsMap = new Hashtable();
      XSLTEngine xslt = new XSLTEngine();
      if (showLog) {
       System.out.println("Processing arguments...");
       System.out.flush();
      }
      createParameterHashtable(argsMap, args);
      if (showLog) {
       System.out.println("Building I/O pointers...");
       System.out.flush();
      }
      FileReader fXML = new FileReader(args[0]); //this is the XML file
      FileReader fXSL = new FileReader(args[1]); //this is the XSL file
      FileWriter fXMLo = new FileWriter(args[2]); //this is the XML file (out)
      if (showLog) {
       System.out.println("Transforming XML...");
       System.out.flush();
      }
      xslt.transformXSL(fXML, fXSL, fXMLo, argsMap);
      if (doCleanup) {
       if (showLog) {
         System.out.println("Cleaning up...");
```

```
        System.out.flush();
      }
      xslt = null;
      fXML = null;
      fXSL = null;
      fXMLo = null;
      argsMap = null;
      System.gc();
      if (showLog) {
        System.out.println("Done");
        System.out.flush();
      }
     }
    }
    catch (Throwable th) {
      th.printStackTrace(System.out);
    }
  }
  private static void createParameterHashtable(Hashtable table,
                                               String[] args)
      throws ARCOLiteException {
    table.clear();
    if((args.length-3) % 2 != 0) {
    //must be even number of arguments to form pair but the first
    //three are special cases for file specifications
      throw new ARCOLiteException(
        new NullPointerException(
              "Parameter map cannot be constructed without " +
                    "a proper set of key-value pairs in the arguments."));
    }
    else {
      //Proceed as normal
      for(int i=3; i<args.length; i+=2) {
       //Enter this loop if args.length > 3
       table.put(args[i],args[i+1]);
       //Debug information
       System.out.println("\tParameter map entry: " +
                        args[i] + " = " + args[i+1]);
       System.out.flush();
      }
    }
  }
}
```

9.1.1.9 XSLTEngine.java

```
package migration;

/**
 * <p>Title: ARCO Platform Migration Tool</p>
 * <p>Description: Tool for migrating between ARCO platforms.</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: University of Sussex, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0 (Build: November 2003)
 */

import java.io.*;
import java.util.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;


class XSLTEngine {
  private TransformerFactory tFactory;
  private ARCOLiteXSLTErrorListener errListener;
  private Transformer engine;
    XSLTEngine() {
      tFactory = TransformerFactory.newInstance();
      errListener = new ARCOLiteXSLTErrorListener();
    }
    /**
     * <p>
     * This is one-step method for transforming an XML file using an
     * XSL template and a set of parameters that can be passed to the
     * XSL template.
     * </p>
     * @param xmlSource java.io.Reader object specifying the source
     * of the XML document on which the transformation is to be applied.
     * @param xslTemplate java.io.Reader object specifying the source
     * of the XSL template which will be used in the transformation.
     * @param result java.io.Writer object specifying the destination
     * to which the result of transformation is written.
     * @param parameters java.util.Hashtable object containing a key-value
     * pair of parameters that can be passed to the XSL template. This
     * is optional. To disable it, use a null value.
     * @throws ARCOLiteException Generalised exception for the ARCOLite
     * architecture.
     */ public void transformXSL(Reader xmlSource, Reader xslTemplate,
                              Writer result, Hashtable parameters)
      throws ARCOLiteException {
      StreamSource xml, xsl;
      StreamResult out;
      if (xmlSource != null) {
       xml = new StreamSource(xmlSource);
      }
      else {
       throw new ARCOLiteException(new Throwable("Invalid XML source."));
      }
      if (xslTemplate != null) {
       xsl = new StreamSource(xslTemplate);
      }
      else {
       throw new ARCOLiteException(new Throwable("Invalid XSL template."));
      }
      if (result != null) {
       out = new StreamResult(result);
      }
      else {
       throw new ARCOLiteException(new Throwable("Invalid XSLT destination."));
      }
      try {
       engine = tFactory.newTransformer(xsl);
       engine.setErrorListener(errListener);
       if (parameters != null) {
         setTransformerParams(parameters);
       }
       engine.transform(xml, out);
       engine.clearParameters();
      }
      catch (Throwable t) {
       throw new ARCOLiteException(t);
      }
    }

    private void setTransformerParams(Hashtable paramSet) {
      Enumeration paramKeys = paramSet.keys();
      while (paramKeys.hasMoreElements()) {
       String paramName = (String) paramKeys.nextElement();
       String paramVal = (String) paramSet.get(paramName);
       if (paramVal != null) {
         engine.setParameter(paramName, paramVal);
       }
      }
    }
}
```

9.1.2 AXTE – ARCOLite XML Transformation Engine

9.1.2.1 AXTEException.java

```
package axte;


/**
 * <p>Title: ARCO XML Transformation Engine (AXTE)</p>
 * <p>Description: Servlets for XML transformation</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: UoS, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0
 */

/**
 * This class is a generalized exception for the package
 * to represent all exceptions that are generated
 * internally. Internally, all exceptions are caught and
 * rethrown as AXTEException.
 *
 * The GUI application is supposed to catch only
 * AXTEExceptions from any class in the xdecelogic package
 * because no other types of exceptions will be thrown from
 * these classes. The GUI application should be able
 * to interpret the exception and either terminate or show
 * error messages to the user depending on the fatality of
 * the exception.
 */
class AXTEException
    extends Throwable
{
  private Throwable theCause;
  AXTEException(Throwable cause) {
    /*
      Maintains an internal Throwable object
     instead of normal subclassing of the Throwable
     class.
     */
    theCause = cause;
  }
  /**
   * Overrides the superclass method and redefines
   * it in the context of this exception class.
   *
   * @return returns the underlying Throwable object
   * that caused the exception.
   */
  public Throwable getCause() {
    return theCause;
  }
  /**
   * Similar to its superclass method, this method
   * prints the stack trace for the underlying Throwable
   * object into System.out.
   */
  public void printStackTrace() {
    theCause.printStackTrace();
  }
  /**
   * Similar to its superclass method, this method
   * prints the stack trace for the underlying Throwable
   * object through the specified PrintWriter object.
   *
   * @param writer java.io.PrintWriter object through which
   * the stack trace is printed.
   */
  public void printStackTrace(java.io.PrintWriter writer) {
    theCause.printStackTrace(writer);
  }
  /**
   * Similar to its superclass method, this method
   * prints the stack trace for the underlying Throwable
   * object into the specified print stream.
   *
   * @param stream java.io.PrintStream object into which
   * the stack trace is serialized.
   */
  public void printStackTrace(java.io.PrintStream stream) {
    theCause.printStackTrace(stream);
  }
  /**
   * Obtains the stack trace as an array of StackTraceElement
   * objects from the underlying Throwable object.
   *
   * @return array of StackTraceElement objects containing
   * all the information regarding the stack trace.
   */
  public StackTraceElement[] getStackTrace() {
    return theCause.getStackTrace();
  }
  /**
   * Similar to all exception classes, this method obtains
   * the descriptive message from the underlying Throwable
   * object.
   *
   * @return String object containing the descriptive message
   * for the exception.
   */
  public String getMessage() {
    return theCause.getMessage();
  }
}
```

9.1.2.2 AXTESAXErrorHandler.java

```
package axte;


/**
 * <p>Title: ARCO XML Transformation Engine (AXTE)</p>
 * <p>Description: Servlets for XML transformation</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: UoS, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0
 */

import java.io.*;

import org.xml.sax.*;


/**
 * <P>
 * <STRONG>AXTESAXErrorHandler</STRONG> implements the
 * org.xml.sax.ErrorHandler interface for use with the built
 * in DOM parser of this package. This class uses the error
 * handling rules as specified in W3C XML 1.0 specifications.
 * Messages are printed out to output stream. At a later stage, RMI
 * based remote server logging may be introduced.
 * </P>
 */
class AXTESAXErrorHandler
    implements ErrorHandler
{
  private boolean strict = false;
  private PrintWriter oStream;
  public AXTESAXErrorHandler() {
    oStream = new PrintWriter(System.out);
  }
  public AXTESAXErrorHandler(Writer output) {
    oStream = new PrintWriter(output);
  }
  /**
   * According to W3C XML 1.0 specifications, this method prints
   * out the information for a non-fatal error to the output stream and
   * lets the parser continue. If the error handler is in less
   * strict mode, this message is not printed to output stream.
   *
   * @param exception the SAXParseException object that is reported
   * by the DOM parser.
   */
  public void error(SAXParseException exception) {
    if (!strict) {
      return; //less-strict mode
    }
    oStream.println("[SAX error]: " + exception.getMessage());
    oStream.flush();
  }
  /**
   * According to the W3C XML 1.0 specifications, this method should
   * stop the parser from parsing. It does so by throwing an exception
   * and not printing out a fatal error message to the output stream.
   *
   * @param exception the SAXParseException object that is reported
   * by the DOM parser
   * @throws SAXException the SAXException thrown to stop the parser
   * from parsing.
   */
  public void fatalError(SAXParseException exception)
      throws SAXException {
    oStream.println("[SAX fatal-error]: " + exception.getMessage());
    oStream.println("Parse process halted");
    oStream.flush();
    throw new SAXException(exception);
  }
  /**
   * According to the W3C XML 1.0 specifications, this method prints out
   * the information for a parser warning to the output stream and lets the
   * parser continue. If the error handler is in less
   * strict mode, this message is not printed to output stream.
   *
   * @param exception the SAXParseException object that is reported by
   * the DOM parser.
   */
  public void warning(SAXParseException exception) {
    if (!strict) {
      return; //less-strict mode
    }
    oStream.println("[SAX warning]: " + exception.getMessage());
    oStream.flush();
  }
  /**
   * This method is used to toggle the strictness of error
   * handling. By default the error handler is in less
   * strict mode and only reports fatal errors when parsing
   * is aborted.
   *
   * @param flag a boolean flag to indicate if the error handler
   * should be strict.
   */
  public void makeStrict(boolean flag) {
    strict = flag;
  }
}
```

9.1.2.3 AXTEXMLReader.java

```
package axte;
```

```
/**
 * <p>Title: ARCO XML Transformation Engine (AXTE)</p>
 * <p>Description: Servlets for XML transformation</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: UoS, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0
 */

import java.io.*;
import javax.xml.parsers.*;

import org.w3c.dom.*;
import org.xml.sax.*;


/**
 * <P><CODE>RawXMLReader</CODE> is a class used by the
 * ARCO XML processor superstructure built to process the
 * XDE or metadata instances. This class lets the programmer
 * obtain the raw XML document using DOM. It is not to be
 * used by the GUI classes. This class is just used to isolate
 * the XML parsing code that is used by several other classes
 * in this package.</P>
 * <P>The complementary class of this class is <CODE>
 * RawXMLWriter</CODE>, which writes a DOM document into
 * an output stream or reader.
 * </P>
 * @author Anirban Basu (ab25@sussex.ac.uk)
 * @version 1.0.0.0
 */

class AXTEXMLReader
{
  private DocumentBuilderFactory myFactory;
  private DocumentBuilder myBuilder;
  private Document myDOMDocument;
  private Node theRoot;
  private AXTESAXErrorHandler errorHandler;

  public AXTEXMLReader() {
    //No need to write Javadoc comments for this constructor
    myFactory = myFactory.newInstance();
    errorHandler = new AXTESAXErrorHandler();
  }
  /**
   * <P>
   * As the name suggests, this method is called before parsing
   * an XML document. The parser settings may be changed every time
   * before the document is parsed.
   * </P>
   *
   * @param expandEntityReferences boolean flag indicating
   * if entity references should be expanded and presented in
   * the DOM tree.
   * @param ignoreComments boolean flag indicating if comment
   * nodes should be ignored by the parser.
   * @param namespaceAware boolean flag indicating if the parser
   * should be namespace considerate.
   * @param validateWhileParsing boolean flag indicating if the
   * parser should validate the XML file. This is just the well-
   * formedness validation. It does not validate against the
   * schema.
   * @param strictErrorReporting boolean flag indicating if the
   * parser should report warnings and non-fatal errors.
   * @throws AXTEException a generalized exception for
   * this package if there is any problem with pre-setting the
   * parser.
   */
  protected void preParseInit(boolean expandEntityReferences,
                            boolean ignoreComments,
                            boolean namespaceAware,
                            boolean validateWhileParsing,
                            boolean strictErrorReporting)
      throws AXTEException {
    try {
      myFactory.setExpandEntityReferences(expandEntityReferences);
      myFactory.setIgnoringComments(ignoreComments);
      myFactory.setNamespaceAware(namespaceAware);
      myFactory.setValidating(validateWhileParsing);
      myBuilder = myFactory.newDocumentBuilder();
      errorHandler.makeStrict(strictErrorReporting);
      myBuilder.setErrorHandler(errorHandler);
    }
    catch (Throwable e) {
      throw new AXTEException(e);
    }
  }

  /**
   * <P>
   * This method parses an XML file based on the parser
   * configuration set by preParseInit method.
   * </P>
   *
   * @param xmlLocation an Object describing the location of
   * the XML file to be parsed. This can be an URI expressed
   * in the form of a String or a java.io.File object or a
   * java.io.InputStream object or org.xml.sax.InputSource
   * object.
   * @throws AXTEException a generalized exception for
   * this package thrown if there is any problem with parsing.
   */
  protected void parseAndPostProcess(Object xmlLocation)
      throws AXTEException {
    /*
     The use of instanceof to determine the class of
     an object at runtime is similar to Runtime Type
     Identification (RTTI) in C++.
     */
    try {
      if (xmlLocation instanceof String) {
        //The location is an URI
        myDOMDocument = myBuilder.parse( (String) xmlLocation);
      }
      else if (xmlLocation instanceof File) {
        myDOMDocument = myBuilder.parse( (File) xmlLocation);
      }
      else if (xmlLocation instanceof InputSource) {
        myDOMDocument = myBuilder.parse( (InputSource) xmlLocation);
      }
      else if (xmlLocation instanceof InputStream) {
        myDOMDocument = myBuilder.parse( (InputStream) xmlLocation);
      }
      else {
        //Nothing else is allowed. Exception should be thrown.
        throw new AXTEException(
            new Throwable("Unknown XML input source type."));
      }
      //Get the top-level element to form the DOM tree
      theRoot = (Node) myDOMDocument;
    }
    catch (Throwable e) {
      throw new AXTEException(e);
    }
  }
  /**
   * <P>
   * This method returns the underlying DOM document
   * created through parsing an XML file.
   * </P>
   *
   * @return org.w3c.dom.Document object that contains
   * information about the parsed XML file.
   */
  protected Document getDOMDocument() {
    return myDOMDocument;
  }
  /**
   * <P>
   * This method returns the top-level node (root) of
   * the DOM tree created through parsing an XML file.
   * </P>
   *
   * @return org.w3c.dom.Node object that is the root
   * of the DOM tree.
   */
  protected Node getTheRootOfDOMTree() {
    return theRoot;
  }

}
```

9.1.2.4 *ContentServ.java (the servlet)*

```
package axte;


import java.io.*;
import java.net.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.xml.transform.*;
import javax.xml.transform.stream.*;
import java.text.*;


/**
 * <p>Title: ARCO XML Transformation Engine (AXTE)</p>
 * <p>Description: Servlets for XML transformation</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: UoS, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0
 */

public class ContentServ
    extends HttpServlet
{
  //Global constants
  private final static String ID_PRESENTATION = "pID";
  private final static String ID_STYLESHEET = "sID";
  private final static String ID_IRF = "irfID";
  private final static String ID_MODE = "modeID"; //changes modes of the
servlet
  private final static String VAR_PRESENTATION = "presentation";
  private final static String FILE_SEPARATOR = System.getProperty(
      "file.separator");

  private final static int MODE_XFORM = 1; //transformation mode
  private final static int MODE_BINMED = 2; //binary mode for media files
  private final static int MODE_BINTPL = 3; //binary mode for templates
  private final static int MODE_XFORM_PLAIN = 4; //simple xml transformation

  private PresentationReader presReader;
  private Presentation currentPresentation;
  private String servletContext;

  public void init()
      throws ServletException {
    presReader = new PresentationReader();
  }
  //Process the HTTP Get request
  public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    //response.setContentType(CONTENT_TYPE); //not necessary
    int modeID = 0;
    String presentationContext, stylesheetContext;
    if (request.getParameter(ID_MODE) != null) {
      modeID = Integer.parseInt(request.getParameter(ID_MODE));
    }
    try {
      servletContext = getServletContext().getRealPath("") + FILE_SEPARATOR;
      if (modeID != MODE_XFORM_PLAIN) {
        if (request.getSession().getAttribute(VAR_PRESENTATION) != null) {
          currentPresentation = (Presentation) request.getSession().
              getAttribute(
              VAR_PRESENTATION);
        }
        else {
          currentPresentation = null;
        }
        processRequestParameters(request);
        //update it again
        currentPresentation = (Presentation)
request.getSession().getAttribute(
            VAR_PRESENTATION);
        presentationContext = servletContext + currentPresentation.path +
            FILE_SEPARATOR;
        stylesheetContext = servletContext +
            currentPresentation.stylesheetPath + FILE_SEPARATOR;
      }
      else {
        currentPresentation = null; //no need, expect source & style
        presentationContext = stylesheetContext = servletContext;
      }
      if (modeID == MODE_XFORM ||
          modeID == MODE_XFORM_PLAIN) { //mode transformation of XML
        PrintWriter out = response.getWriter(); //we need character stream
        TransformerFactory tFactory =
            TransformerFactory.newInstance();
        Source xmlSource = null;
        Source xslSource = null;
        Transformer transformer = null;
        // Get params from URL.
        String requestSource = request.getParameter("source");
        // Get the XML input document.
        if (requestSource != null && requestSource.length() > 0) {
          xmlSource = new StreamSource(new URL("file", "",
                                               presentationContext +
                                               requestSource).
                                       openStream());
        }
        else {
          xmlSource = new StreamSource(new URL("file", "",
                                               presentationContext +
                                               currentPresentation.mainData).
                                       openStream());
        }
        // Get the stylesheet.
        String requestStyle = request.getParameter("style");
        if (requestStyle != null && requestStyle.length() > 0) {
          xslSource = new StreamSource(new URL("file", "",
                                               stylesheetContext +
                                               requestStyle).openStream());
        }
        else {
          xslSource = new StreamSource(new URL("file", "",
                                               stylesheetContext +

currentPresentation.styleSheet).
                                       openStream());
        }
        if (xmlSource != null) { // We have an XML input document.
          if (xslSource == null) { // If no stylesheet, look for PI
            String media = null, title = null, charset = null;
            xslSource = tFactory.getAssociatedStylesheet(xmlSource, media,
                                                         title, charset);
          }
          if (xslSource != null) { // Now do we have a stylesheet?
            transformer = tFactory.newTransformer(xslSource);
            transformer.clearParameters();
            //Add special parameters
            //these two will be over written if necessary
            if (modeID != MODE_XFORM_PLAIN) {
              transformer.setParameter("source",
                                       (String) currentPresentation.mainData);
              transformer.setParameter("style",
                                       (String)
currentPresentation.styleSheet);
            }
            else {
              transformer.setParameter("source", requestSource);
              transformer.setParameter("style", requestStyle);
            }
            transformer.setParameter("contextPath",
                                     (String) presentationContext);
            transformer.setParameter("host", request.getServerName());
            transformer.setParameter("port", "" + request.getServerPort() +
"");
            transformer.setParameter("servletPath",
                                     request.getContextPath() +
                                     request.getServletPath());
            // Set stylesheet params.
            setTransformerParameters(transformer, request);
            //use cache-control
            addCacheExpirationModel(response);
            // Perform the transformation.
            transformer.transform(xmlSource, new StreamResult(out));
          }
          else {
            throw new AXTEException(new Throwable(
                "Invalid stylesheet specification."));
          }
        }
        else {
          throw new AXTEException(new Throwable("Invalid source XML (" +
                                                presentationContext +
                                                requestSource +
                                                ")"));
        }
        out.close();
      }
      else if (modeID == MODE_BINMED) { //mode binary stream serve for media
        ServletOutputStream out = response.getOutputStream();
        //response.setContentType(request.getParameter("mime"));
        String mediaFile = request.getParameter("filename");
        if (mediaFile == null ||
            mediaFile.length() <= 0) {
          throw new AXTEException(
              new Throwable("No media file specified."));
        }
        FileInputStream fIn = new FileInputStream(
            presentationContext + mediaFile);
        if (request.getParameter("cache") == null) {
          //use cache-control
          addCacheExpirationModel(response);
        }
        //or use stylesheet context path or a different modeID
        sendFileToOutput(fIn, out);
        fIn.close();
        out.close();
      }
      else if (modeID == MODE_BINTPL) { //mode binary stream serve for
template
        ServletOutputStream out = response.getOutputStream();
        String mediaFile = request.getParameter("filename");
        if (mediaFile == null ||
            mediaFile.length() <= 0) {
          throw new AXTEException(
              new Throwable("No binary input stream specified."));
        }
        FileInputStream fIn = new FileInputStream(
            stylesheetContext + mediaFile);
        //use cache-control
        addCacheExpirationModel(response);
        //or use stylesheet context path or a different modeID
        sendFileToOutput(fIn, out);
        fIn.close();
        out.close();
      }
      else {
        //undefined mode of operation
        throw new AXTEException(
            new Throwable("Servlet mode not implemented."));
      }
    }
    catch (Throwable e) {
      getServletContext().log("AXTE exception", e);
      response.sendError(response.SC_INTERNAL_SERVER_ERROR, e.getMessage());
      //e.printStackTrace(); //for debugging only
    }
  }
  //Clean up resources
  public void destroy() {
    presReader = null;
  }

  private boolean processRequestParameters(HttpServletRequest request)
      throws Throwable {
    //first check session variable availability
    if (currentPresentation == null) {
      //session expired or new session, so proceed
      if (request.getParameter(ID_PRESENTATION) == null ||
          request.getParameter(ID_STYLESHEET) == null ||
          request.getParameter(ID_IRF) == null) {
        throw new AXTEException(new Throwable(
            "Session expired and invalid request parameters."));
      }
      else {
        presReader.loadIRF(servletContext + request.getParameter(ID_IRF) +
                           ".xml");
        presReader.loadPresentation(request.getParameter(ID_PRESENTATION),
                                    request.getParameter(ID_STYLESHEET));
        request.getSession().setAttribute(VAR_PRESENTATION,
                                          presReader.loadedPresentation);
        return true;
      }
    }
    else {
      if (request.getParameter(ID_PRESENTATION) == null ||
          request.getParameter(ID_STYLESHEET) == null ||
          request.getParameter(ID_IRF) == null) {
        return false;
      }
      if (Integer.parseInt(request.getParameter(ID_PRESENTATION)) !=
          currentPresentation.pID ||
          Integer.parseInt(request.getParameter(ID_STYLESHEET)) !=
          currentPresentation.sID) {
        presReader.loadIRF(servletContext + request.getParameter(ID_IRF) +
                           ".xml");
        presReader.loadPresentation(request.getParameter(ID_PRESENTATION),
                                    request.getParameter(ID_STYLESHEET));
        if (presReader.loadedPresentation == null) {
          throw new AXTEException(
              new Throwable("Invalid presentation parameters."));
        }
        request.getSession().setAttribute(VAR_PRESENTATION,
                                          presReader.loadedPresentation);
        return true;
      }
      else {
        return false;
      }
    }
  }

  private void setTransformerParameters(Transformer transformer,
                                        HttpServletRequest request)
      throws Exception {
    Enumeration paramNames = request.getParameterNames();
    while (paramNames.hasMoreElements()) {
      String paramName = (String) paramNames.nextElement();
      String paramVal = request.getParameter(paramName);
      if (paramVal != null) {
        if (paramVal.length() > 0) {
          transformer.setParameter(paramName, paramVal);
        }
      }
    }
  }

  private void sendFileToOutput(FileInputStream fInS,
                                OutputStream outS)
      throws Throwable {
    while (true) {
      int nReadSize = fInS.available();
      if (nReadSize <= 0) {
        break;
      }
      byte[] buffer = new byte[nReadSize];
      fInS.read(buffer);
      outS.write(buffer);
      outS.flush(); //so that something is sent in every iteration
    }
  }

  private void addCacheExpirationModel(HttpServletResponse response) {
    Calendar cal = Calendar.getInstance(TimeZone.getDefault());
    SimpleDateFormat sdf = new SimpleDateFormat(
        "EEE, dd MMM yyyy hh:mm:ss z");
    //refer to HTTP 1.1 RFC 2068 sections 13, 14, etc.
    response.setHeader("Cache-Control",
                       "no-cache, no-store, must-revalidate, proxy-
revalidate");
    /*response.setHeader("Cache-Control",
                       "post-check=0, pre-check=0");*/
    //non standard header
    response.setHeader("Pragma", "no-cache");
    //more to be sure
    response.setHeader("Expires", sdf.format(cal.getTime()));
    response.setHeader("Last-Modified", sdf.format(cal.getTime()));
  }

}
```

9.1.2.5 Presentation.java

```
package axte;


/**
 * <p>Title: ARCO XML Transformation Engine (AXTE)</p>
 * <p>Description: Servlets for XML transformation</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: UoS, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0
 */

class Presentation
{
  //this class is very much like a structure
  protected int pID, sID; //the requested presentation ID and style sheet ID
  protected String path; //this is presentation directory
  protected String mainData; //PLITE instance
  protected String stylesheetPath; //path of the stylesheet directory
  protected String styleSheet; //associated main style sheet
}
```

9.1.2.6 PresentationReader.java

```
package axte;


/**
 * <p>Title: ARCO XML Transformation Engine (AXTE)</p>
 * <p>Description: Servlets for XML transformation</p>
 * <p>Copyright: Copyright (c) 2003, ARCO Consortium. All rights reserved.</p>
 * <p>Company: UoS, ARCO Consortium</p>
 * @author Anirban Basu, UoS (ab25@sussex.ac.uk)
 * @version 1.0
 */

import java.io.*;

import org.apache.xpath.*;
import org.w3c.dom.*;


class PresentationReader
{
  private AXTEXMLReader xmlReader;
  private Node rootNode;
  /**
   * Protected presentation object that is used to store the
   * read in presentation data.
   */
  protected Presentation loadedPresentation;
  /**
   * <P>Non-parameterized constructor of the class.</P>
   */
  PresentationReader() {
    xmlReader = new AXTEXMLReader();
  }
  /**
   * <P>
   * This method performs a recursive search on the given node (startHere)
   * that matches the name used for the search.
   * </P>
   * @param elementName String object specifying the name of the element
   * to search.
   * @param startHere org.w3c.dom.Node object specifying the node on which
   * the recursive search is performed.
   * @return org.w3c.dom.Node object returning the matching element; null if
   * not found.
   */
  protected Node findElement(String elementName,
                             Node startHere) {
    NodeList nList = startHere.getChildNodes();
    for (int i = 0; i < nList.getLength(); i++) {
      if (nList.item(i).getNodeName().compareTo(elementName) == 0 //match name
          && nList.item(i).getNodeType() == Node.ELEMENT_NODE) { //match type
        return nList.item(i);
      }
      else {
        Node depthSearch = findElement(elementName, nList.item(i));
        if (depthSearch != null) {
          return depthSearch;
        }
      }
    }
    return null;
  }

  protected Node findElementUsingXPath(String xpathExpression,
                                       Node startHere)
      throws AXTEException {
    try {
      return XPathAPI.selectSingleNode(startHere, xpathExpression);
    }
    catch (Throwable e) {
      throw new AXTEException(e);
    }
  }

  /**
   * <P>This method gets the value text associated with the given Node
   * assuming that it is an element.</P>
   * @param element org.w3c.dom.Node object specifying the element whose
   * value is to be returned.
   * @return String object containing the value of the element, null if
   * the value is unavailable.
   */
  protected String getValueOfElement(Node element) {
    NodeList nl = element.getChildNodes();
    for (int i = 0; i < nl.getLength(); i++) {
      if (nl.item(i).getNodeType() == Node.TEXT_NODE) {
        return nl.item(i).getNodeValue();
      }
    }
    return null;
  }

  protected void loadIRF(String irfPath)
      throws AXTEException {
    xmlReader.preParseInit(false, true, false, false, false);
    xmlReader.parseAndPostProcess(new File(irfPath));
    rootNode = xmlReader.getTheRootOfDOMTree();
  }

  protected void loadPresentation(String presentationID,
                                  String stylesheetID)
      throws Throwable {
    //Could use XPATH through XPathAPI
    loadedPresentation = new Presentation();
    Element firstPresentation = (Element) findElement(
        "PRESENTATION", rootNode);
    if (firstPresentation == null) {
      throw new AXTEException(
          new Throwable("Invalid presentation packaging."));
    }
    Node currentPresentation = firstPresentation;
    while (true) {
      if (currentPresentation == null) {
        break;
      }
      Node nID = findElement("ID", currentPresentation);
      if (nID != null) {
        if (getValueOfElement(nID).compareTo(presentationID) == 0) {
          loadedPresentation.pID = Integer.parseInt(presentationID);
          loadedPresentation.mainData = getValueOfElement(
              findElement("MAIN_DATA", nID.getParentNode()));
          loadedPresentation.path = getValueOfElement(
              findElement("DATA_PATH", nID.getParentNode()));
          loadedPresentation.sID = Integer.parseInt(stylesheetID);
          getStylesheet(stylesheetID);
          break;
        }
      }
      currentPresentation = currentPresentation.getNextSibling();
      if (currentPresentation.getNodeName().compareTo("STYLESHEET") == 0) {
        break;
      }
    }
    if ( (loadedPresentation.pID == 0)
        || (loadedPresentation.mainData == null)
        || (loadedPresentation.path == null)) {
      loadedPresentation = null; //nullify
      throw new AXTEException(
          new Throwable("Requested presentation does not exist."));
    }
    if (loadedPresentation.styleSheet == null) {
      loadedPresentation = null; //nullify
      throw new AXTEException(
          new Throwable("Requested stylesheet does not exist."));
    }
  }
  private void getStylesheet(String stylesheetID)
      throws AXTEException {
    Element firstStylesheet = (Element) findElement(
        "STYLESHEET", rootNode);
    if (firstStylesheet == null) {
      throw new AXTEException(
          new Throwable("Invalid stylesheet packaging."));
    }
    Node currentStylesheet = firstStylesheet;
    while (currentStylesheet != null) {
      Node nID = findElement("ID", currentStylesheet);
      if (nID != null) {
        if (getValueOfElement(nID).compareTo(stylesheetID) == 0) {
          loadedPresentation.stylesheetPath = getValueOfElement(
              findElement("STYLESHEET_PATH", nID.getParentNode()));
          loadedPresentation.styleSheet = getValueOfElement(
              findElement("MAIN_STYLE", nID.getParentNode()));
          return;
        }
      }
      currentStylesheet = currentStylesheet.getNextSibling();
    }
    loadedPresentation.styleSheet = null;
  }
}
```

9.2 C, C++ code

9.2.1 Communication between exhibition server and ARIFLite

9.2.1.1 File: MSXML4Wrapper.h

```
// MSXML4Wrapper.h: interface for the CMSXML4Wrapper class.
//
//////////////////////////////////////////////////////////////////////

#if
!defined(AFX_MSXML4WRAPPER_H__1877B15E_CFEC_4DDA_96E9_B8A8F0617B12__INCLUDED_)
#define AFX_MSXML4WRAPPER_H__1877B15E_CFEC_4DDA_96E9_B8A8F0617B12__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#import <msxml4.dll>
using namespace MSXML2;
```

```
class CMSXML4Wrapper
{
private:
	IXMLDOMDocument2Ptr pXMLDom, pXSLStyle, pXMLOut;
	IXSLTemplatePtr pXSLTemplate;
	IXSLProcessorPtr pXSLTEngine;
	HRESULT hr;
	bool isLoaded;
public:
	CMSXML4Wrapper();
	virtual ~CMSXML4Wrapper();
	BOOL InitLibrary();
	BOOL LoadXMLFile(CString FilePath);
	void UnloadXML();
	BOOL SaveXMLFile(CString FilePath, IXMLDOMDocument2Ptr pDoc = NULL);
	IXMLDOMDocument2Ptr GetXMLDOMDoc();
	IXMLDOMNodePtr GetDOMNode(CString XPathExpression, IXMLDOMNodePtr
pStartHere = NULL);
	IXMLDOMNodeListPtr GetDOMNodes(CString XPathExpression, IXMLDOMNodePtr
pStartHere = NULL);

	BOOL PrepareXSLT(CString XSLStylesheetPath, IXMLDOMDocument2Ptr pDoc =
NULL);
	BOOL AddXSLParameters(CString paramName, CString paramValue);
	BOOL ApplyXSLT(CString XMLOutputPath);
};

#endif //
!defined(AFX_MSXML4WRAPPER_H__1877B15E_CFEC_4DDA_96E9_B8A8F0617B12__INCLUDED_)
```

9.2.1.2 File: MSXML4Wrapper.cpp

```
// MSXML4Wrapper.cpp: implementation of the CMSXML4Wrapper class.
//
//////////////////////////////////////////////////////////////////////

#include "stdafx.h"
#include "ariflite.h"
#include "MSXML4Wrapper.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

//////////////////////////////////////////////////////////////////////
// Construction/Destruction
//////////////////////////////////////////////////////////////////////

CMSXML4Wrapper::CMSXML4Wrapper()
{
	isLoaded = false;
	CoInitialize(NULL);
}

CMSXML4Wrapper::~CMSXML4Wrapper()
{
	pXSLStyle.Release();
	pXMLOut.Release();
	pXSLTemplate.Release();
	CoUninitialize();
}

BOOL CMSXML4Wrapper::InitLibrary()
{
	hr = pXMLDom.CreateInstance(__uuidof(DOMDocument40));
	if(FAILED(hr))
		return FALSE;
	pXMLDom->async = VARIANT_FALSE;
	hr = pXSLStyle.CreateInstance(__uuidof(FreeThreadedDOMDocument40));
	if(FAILED(hr))
		return FALSE;
	pXSLStyle->async = VARIANT_FALSE;

	hr = pXMLOut.CreateInstance(__uuidof(DOMDocument40));
	if(FAILED(hr))
		return FALSE;
	pXMLOut->async = VARIANT_FALSE;

	hr = pXSLTemplate.CreateInstance(__uuidof(XSLTemplate40));
	if(FAILED(hr))
		return FALSE;

	return TRUE;
}

BOOL CMSXML4Wrapper::LoadXMLFile(CString FilePath)
{
	if(pXMLDom->load((LPCTSTR)FilePath)==VARIANT_TRUE)
	{
		isLoaded = true;
		return TRUE;
	}
	else
		return FALSE;
}

void CMSXML4Wrapper::UnloadXML()
{
	if(isLoaded)
		pXMLDom.Release();
	isLoaded = false;
}

BOOL CMSXML4Wrapper::SaveXMLFile(CString FilePath, IXMLDOMDocument2Ptr pDoc)
{
	if(pDoc == NULL)
	{
		if(isLoaded)
		{
			hr = pXMLDom->save((LPCTSTR)FilePath);
			if(FAILED(hr))
				return FALSE;
			else
				return TRUE;
		}
		else
			return FALSE;
	}
	else
	{
		hr = pDoc->save((LPCTSTR)FilePath);
		if(FAILED(hr))
			return FALSE;
		else
			return TRUE;
	}
}

IXMLDOMDocument2Ptr CMSXML4Wrapper::GetXMLDOMDoc()
{
	if(isLoaded)
		return pXMLDom;
	else
		return NULL;
}

IXMLDOMNodePtr CMSXML4Wrapper::GetDOMNode(CString XPathExpression,
IXMLDOMNodePtr pStartHere)
{
        _bstr_t query = XPathExpression;
        if(!isLoaded)
              return NULL;
        IXMLDOMNodePtr pNode;
        if(pStartHere == NULL)
        {
              //start at root
              pNode = pXMLDom->selectSingleNode(query);
        }
        else
        {
              //start at given node
              pNode = pStartHere->selectSingleNode(query);
        }
        if(pNode == NULL)
              return NULL;
        else
              return pNode;
}

IXMLDOMNodeListPtr CMSXML4Wrapper::GetDOMNodes(CString XPathExpression,
IXMLDOMNodePtr pStartHere)
{
        _bstr_t query = XPathExpression;
        if(!isLoaded)
              return NULL;
        IXMLDOMNodeListPtr pNodes;
        if(pStartHere == NULL)
        {
              //start at root
              pNodes = pXMLDom->selectNodes(query);
        }
        else
        {
              //start at given node
              pNodes = pStartHere->selectNodes(query);
        }
        if(pNodes == NULL)
              return NULL;
        else
              return pNodes;
}

/*
About XML transformation:

  Load the XML file as a IXMLDocument2Ptr. Load the stylesheet
  as another IXMLDocument2Ptr. Create a IXSLTemplatePtr and put
  the stylesheet as reference. Create a IXSLProcessorPtr from the
  template object. Load the XML file as input to this processor.
  Load another IXMLDocument2Ptr as the output. Transform! Save
  the result IXMLDocument2Ptr
*/

BOOL CMSXML4Wrapper::PrepareXSLT(CString XSLStylesheetPath,
IXMLDOMDocument2Ptr pDoc)
{
        if(pXSLStyle->load((LPCTSTR)XSLStylesheetPath)!=VARIANT_TRUE)
              return FALSE;

        pXSLTemplate->PutRefstylesheet(pXSLStyle);

        pXSLTEngine = pXSLTemplate->createProcessor();
        if(pXSLTEngine==NULL)
              return FALSE;

        if(pDoc!=NULL)
        {
              pXSLTEngine->Putinput(_variant_t(pDoc.GetInterfacePtr()));
        }
        else
```

```
	{
		if(isLoaded)
		{
			pXSLTEngine-
>Putinput(_variant_t(pXMLDom.GetInterfacePtr()));
		}
		else
			return FALSE;
	}

	pXSLTEngine->Putoutput(_variant_t(pXMLOut.GetInterfacePtr()));
	return TRUE;
	//do something here to add parameters
}

BOOL CMSXML4Wrapper::AddXSLParameters(CString paramName, CString paramValue)
{
	hr = pXSLTEngine->addParameter(_bstr_t(paramName),
		_variant_t(paramValue),"");
	if(FAILED(hr))
		return FALSE;
	return TRUE;
}

BOOL CMSXML4Wrapper::ApplyXSLT(CString XMLOutputPath)
{
	if(pXSLTEngine->transform()!=VARIANT_TRUE)
		return FALSE;
	SaveXMLFile(XMLOutputPath,pXMLOut);
	return TRUE;
}
```

9.2.1.3 File: ARIFLiteConnectivity.h

```
#define ARPROTO "arif://"
#define HTTPPROTO "http://"
#define IO_CHUNKSIZE 65536 //optimise this number; for hard disks this can be
high but for slow networks?
#define ARIF_HTTP_USER_AGENT "Mozilla/4.0 (compatible; ARIFLite 1.0; Windows
(any); UoS, ARCO Consortium)"
#define ARIF_HTTP_VERSION "HTTP/1.1"
#define MAX_CACHE_ENTRY_INFO_SIZE 4096

//error codes
#define ARERR_SUCCESS 0 //no error
#define ARERR_NONET -1 //no internet connectivity
#define ARERR_NOHOSTCONNECT -2 //no connection to host
#define ARERR_NOHTTPREQUEST -3 //no HTTP request can be created
#define ARERR_NOHTTPREQUESTSENT -4 //HTTP request cannot be sent
#define ARERR_INVALIDURL -5 //invalid HTTP URL

#include <afxinet.h>

#if _MSC_VER >= 1000
#pragma once
#endif // _MSC_VER >= 1000

class CARIFLiteConnectivity : public CObject {
public:
	CARIFLiteConnectivity();
	BOOL RetrieveResource(char *lpszARUrl, const char *lpszLocalFilePath);
	int GetError();
	char *InterpretError(int nError);
private:
	int nErrorCode;
	CString vrmlData;
	int seekLocation;
	BOOL HandleARUrl(char *lpszARUrl);
	BOOL DownloadResource(char *lpszUrl, const char *lpszDestination,DWORD
flags = INTERNET_FLAG_TRANSFER_BINARY);
	void ReadVrmlFile(const char *lpszLocation);
```

```
        void WriteVrmlFile(const char *lpszLocation);
        void SetTextureUrl(const char *lpszReplace, const char *lpszUrl);
        void CleanupDirectory(const char *lpszSearchDir);
};
```

9.2.1.4 File: ARIFLiteConnectivity.cpp

```
#include "stdafx.h"
#include "arifliteConnectivity.h"
//#include "md5.h"
#include "MSXML4Wrapper.h"

CARIFLiteConnectivity::CARIFLiteConnectivity()
{
        //count_loading = 0;
        nErrorCode = ARERR_SUCCESS;
}

int CARIFLiteConnectivity::GetError()
{
        return nErrorCode;
}

char *CARIFLiteConnectivity::InterpretError(int nError)
{
        switch(nError)
        {
        case ARERR_SUCCESS:
                return "Success (this is not an error!)";
                break;
        case ARERR_NONET:
                return "No Internet connection found.";
                break;
        case ARERR_NOHOSTCONNECT:
                return "Requested host cannot be connected.";
                break;
        case ARERR_NOHTTPREQUEST:
                return "HTTP request cannot be formed.";
                break;
        case ARERR_NOHTTPREQUESTSENT:
                return "HTTP request cannot be sent.";
                break;
        case ARERR_INVALIDURL:
                return "Cannot decompose URL into component parts.";
                break;
        default:
                return "Unknown error!";
        }
}

BOOL CARIFLiteConnectivity::RetrieveResource(char *lpszARUrl,
                                                                                 const
char *lpszLocalFilePath)
{
/*0
The algorithm: convert the lpszARUrl to HTTP URL by replacing
"arif://" by "http://". If the ARUrl is already "http://", then
do not do anything to it. Use the HTTP URL to locate the
local copy of the resource in the browser cache. If found
in cache, then copy that file using binary stream I/O to
the lpszLocalFilePath. If there is a cache miss, then use WinInet
to download the resource to the browser cache and copy it
to the LocalFilePath.
        */
        nErrorCode = ARERR_SUCCESS;
        if(HandleARUrl(lpszARUrl)==FALSE) return FALSE;
        CString vrmlDir = "Wrl\\";
        CleanupDirectory((LPCTSTR)vrmlDir);
        //download the ARDATA.XML now and then download the rest
        //using cache check but not now.
        CMSXML4Wrapper *p_xmlLib = new CMSXML4Wrapper();
```

```
      p_xmlLib->InitLibrary();
      if(DownloadResource(lpszARUrl,lpszLocalFilePath))
      {
             p_xmlLib->LoadXMLFile(lpszLocalFilePath);
             IXMLDOMNodeListPtr irList = p_xmlLib-
>GetDOMNodes("/AR_DATA/AR_OBJECT");
             for(long ilist=0; ilist<irList->Getlength(); ilist++) {
                    IXMLDOMNodePtr refNode = irList->Getitem(ilist);
                    IXMLDOMNodePtr xmlNode = p_xmlLib-
>GetDOMNode("URL",refNode);
                    if(xmlNode)
                    {
                           //_bstr_t _b_value = xmlNode-
>GetnodeValue().bstrVal;
                           //MessageBox(NULL,(LPCTSTR)(xmlNode-
>Gettext()),"Test",MB_OK);
                           //download the arobject
                           CString vrmlFileName = vrmlDir + "ariflite";
                           char *numbuf = new char[4];
                           memset(numbuf,0,4);
                           ltoa((ilist+1),numbuf,10);
                           vrmlFileName +=numbuf;
                           vrmlFileName+=".wrl";
                           DownloadResource((char *)xmlNode-
>Gettext(),vrmlFileName);
                           //download ar subobjects
                           IXMLDOMNodeListPtr xmlList = p_xmlLib-
>GetDOMNodes("AR_SUBOBJECT/URL",refNode);
                           ReadVrmlFile((LPCTSTR)vrmlFileName);
                           SetTextureUrl(_T("ARCO_DATA//"),
                                  _T("ARCO_DATA/"));

      SetTextureUrl(_T("ContentServ?modeID=2&filename=ARCO_DATA/"),
                                  _T(""));
                           for(long i=0; i<xmlList->Getlength();i++)
                           {
                                  IXMLDOMNodePtr nPtr = xmlList->Getitem(i);
                                  //MessageBox(NULL,nPtr-
>Gettext(),"Test",MB_OK);
                                  CString filename = (char *)nPtr->Gettext();
                                  /*char *bf = new char[4];
                                  memset(bf,0,4);
                                  ltoa(i,bf,10);
                                  filename +=bf;
                                  filename += ".jpg";*/
                                  int find = filename.Find("ARCO_DATA");
                                  filename = filename.Mid(find + 11);
                                  CString textureFile = "Wrl/" + filename +
".jpg";

      SetTextureUrl((LPCTSTR)filename,(LPCTSTR)textureFile);
                                  DownloadResource((char *)nPtr-
>Gettext(),(LPCTSTR)textureFile);
                                  //DownloadResource((char *)nPtr-
>Gettext(),(LPCTSTR)filename);
                                  //filename = " \"" + filename + "\"";
                           }
                           WriteVrmlFile((LPCTSTR)vrmlFileName);
                    }
             }
             p_xmlLib->UnloadXML();
             delete p_xmlLib;
             return TRUE;
      }
      else return FALSE;
}

BOOL CARIFLiteConnectivity::HandleARUrl(char *lpszARUrl)
{
      if(strstr(lpszARUrl,ARPROTO)>0)
      {
```

```
		//replace "arif://" with "http://"
		lpszARUrl[0] = 'h';
		lpszARUrl[1] = 't';
		lpszARUrl[2] = 't';
		lpszARUrl[3] = 'p';
		return TRUE;
	}
	else if(strstr(lpszARUrl,HTTPPROTO)>0)
	{
		//already "http://"
		return TRUE;
	}
	else
	{
		//log error if necessary
		return FALSE; //unknown protocol
	}
}

BOOL CARIFLiteConnectivity::DownloadResource(char *lpszUrl,
                                                                         const
char *lpszDestination,
                                                                         DWORD
flags)
{
	CInternetSession *pSession = new CInternetSession();
	CHttpFile *pFile = NULL;
	char *lpszBuffer = new char[IO_CHUNKSIZE];
	pFile = (CHttpFile *)pSession->OpenURL(lpszUrl,1,flags);
	if(pFile)
	{
		CFile outFile(lpszDestination,
			CFile::typeBinary | CFile::modeWrite | CFile::modeCreate);
		while(true)
		{
			UINT bufferSize = pFile->Read(lpszBuffer,IO_CHUNKSIZE);
			if(bufferSize == 0) break;
			outFile.Write(lpszBuffer,bufferSize);
			outFile.Flush();
			memset(lpszBuffer,0,IO_CHUNKSIZE);
		}
		outFile.Close();
	}
	else
	{
		return FALSE;
	}
	pFile->Close();
	pSession->Close();

	return TRUE;
}

void CARIFLiteConnectivity::ReadVrmlFile(const char *lpszLocation)
{
	CFile vrmlFile;
	vrmlFile.Open(lpszLocation,
		CFile::typeBinary | CFile::modeRead);
	vrmlData = "";
	char *lpszBuffer = new char[IO_CHUNKSIZE];
	while(true)
	{
		memset(lpszBuffer,0,IO_CHUNKSIZE);
		UINT bufferSize = vrmlFile.Read(lpszBuffer,IO_CHUNKSIZE-1);
		if(bufferSize == 0) break;
		vrmlData += lpszBuffer;
	}
	vrmlFile.Close();
	seekLocation = 0;
```

```
}

void CARIFLiteConnectivity::WriteVrmlFile(const char *lpszLocation)
{
	CFile vrmlFile;
	vrmlFile.Open(lpszLocation,
		CFile::typeBinary | CFile::modeWrite | CFile::modeCreate);
	vrmlFile.Write((LPCTSTR)vrmlData,vrmlData.GetLength());
	vrmlFile.Close();

	vrmlData = "";

	//count_loading = 1;
}

void CARIFLiteConnectivity::SetTextureUrl(const char *lpszReplace, const char
*lpszUrl)
{

	vrmlData.Replace(lpszReplace,lpszUrl);
}

void CARIFLiteConnectivity::CleanupDirectory(const char *lpszSearchDir)
{
	WIN32_FIND_DATA FileData;
	HANDLE hFile;
	//clean all jpeg files first
	CString filePattern = lpszSearchDir;
	filePattern +="*.jpg";
	hFile = FindFirstFile((LPCTSTR)filePattern,&FileData);
	if(hFile)
	{
		while(true)
		{
			CString delFile = lpszSearchDir;
			delFile += FileData.cFileName;
			DeleteFile((LPCTSTR)delFile);
			if(FindNextFile(hFile,&FileData)==FALSE) break;
		}
	}
	FindClose(hFile);
	hFile = NULL;
	filePattern = lpszSearchDir;
	filePattern +="*.wrl";
	hFile = FindFirstFile((LPCTSTR)filePattern,&FileData);
	if(hFile)
	{
		while(true)
		{
			CString delFile = lpszSearchDir;
			delFile += FileData.cFileName;
			DeleteFile((LPCTSTR)delFile);
			if(FindNextFile(hFile,&FileData)==FALSE) break;
		}
	}
	FindClose(hFile);
	//clean all vrml files now
}
```

9.3 XSL code

Because of the flexible nature of XSL, the several web style sheets vary from each other in their design but they all follow the same basic rules. Thus, only example XSL code is attached.

9.3.1 Augmented reality and web style sheets

9.3.1.1 File: arserve.xsl (Augmented Reality style sheet)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
      <xsl:output method="xml" encoding="utf-8" indent="yes" omit-xml-
declaration="no"/>
      <xsl:param name="host"/>
      <xsl:param name="port"/>
      <xsl:param name="source"/>
      <xsl:param name="servletPath"/>
      <xsl:param name="contextPath"/>
      <xsl:param name="arMode"/>
      <xsl:param name="arSelect"/>
      <xsl:param name="pID"/>
      <xsl:param name="sID"/>
      <xsl:param name="irfID"/>
      <xsl:template match="/">
            <xsl:element name="AR_DATA">
                  <xsl:choose>
                        <xsl:when test="$arMode='listP'">
                              <xsl:call-template name="listPLITE"/>
                        </xsl:when>
                        <xsl:when test="$arMode='listI'">
                              <xsl:variable name="irFile">
                                    <xsl:value-of select="$contextPath"/>
                                    <xsl:value-of select="$source"/>
                              </xsl:variable>
                              <xsl:call-template name="listIR">
                                    <xsl:with-param name="dataFile">
                                          <xsl:value-of select="$irFile"/>
                                    </xsl:with-param>
                              </xsl:call-template>
                        </xsl:when>
                        <xsl:when test="$arMode='selectI'">
                              <xsl:variable name="irFile">
                                    <xsl:value-of select="$contextPath"/>
                                    <xsl:value-of select="$source"/>
                              </xsl:variable>
                              <xsl:call-template name="listIR">
                                    <xsl:with-param name="dataFile">
                                          <xsl:value-of select="$irFile"/>
                                    </xsl:with-param>
                              </xsl:call-template>
                        </xsl:when>
                  </xsl:choose>
            </xsl:element>
      </xsl:template>
      <xsl:template name="listIR">
            <xsl:param name="dataFile"/>
            <xsl:if test="$arSelect">
                  <xsl:for-each
select="document($dataFile)/INFORMATION_RESOURCE/DATA/MEDIA_OBJECT">
                        <xsl:if test="contains(ENTRY_ID,$arSelect)">
                              <xsl:element name="AR_OBJECT">
                                    <xsl:element
name="URL">http://<xsl:value-of select="$host"/>:<xsl:value-of
select="$port"/>
                                          <xsl:value-of
select="$servletPath"/>?modeID=2&amp;pID=<xsl:value-of
select="$pID"/>&amp;sID=<xsl:value-of select="$sID"/>&amp;irfID=<xsl:value-of
select="$irfID"/>&amp;filename=<xsl:value-of
select="DATA_MEDIA_OBJECT/FILE/FILE_LOCATION"/>
                                    </xsl:element>
                                    <xsl:element name="MIME_TYPE">
                                          <xsl:value-of
select="DATA_MEDIA_OBJECT/FILE/MIME_TYPE"/>
                                    </xsl:element>
```

```
                                        <xsl:for-each
select="DATA_MEDIA_OBJECT/SUB_MEDIA_OBJECT">
                                              <xsl:element
name="AR_SUBOBJECT">
                                                    <xsl:element
name="URL">http://<xsl:value-of select="$host"/>:<xsl:value-of
select="$port"/>
                                                          <xsl:value-of
select="$servletPath"/>?modeID=2&amp;pID=<xsl:value-of
select="$pID"/>&amp;sID=<xsl:value-of select="$sID"/>&amp;irfID=<xsl:value-of
select="$irfID"/>&amp;filename=<xsl:value-of
select="DATA_SUB_MEDIA_OBJECT/FILE/FILE_LOCATION"/>
                                                    </xsl:element>
                                                    <xsl:element
name="MIME_TYPE">
                                                          <xsl:value-of
select="DATA_SUB_MEDIA_OBJECT/FILE/MIME_TYPE"/>
                                                    </xsl:element>
                                              </xsl:element>
                                        </xsl:for-each>
                                  </xsl:element>
                            </xsl:if>
                      </xsl:for-each>
                </xsl:if>
                <xsl:if test="not($arSelect)">
                      <xsl:for-each
select="document($dataFile)/INFORMATION_RESOURCE/DATA/MEDIA_OBJECT">
                            <xsl:if
test="contains(DATA_MEDIA_OBJECT/FILE/MIME_TYPE,'model/vrml')">
                                  <xsl:element name="AR_OBJECT">
                                        <xsl:element
name="URL">http://<xsl:value-of select="$host"/>:<xsl:value-of
select="$port"/>
                                              <xsl:value-of
select="$servletPath"/>?modeID=2&amp;pID=<xsl:value-of
select="$pID"/>&amp;sID=<xsl:value-of select="$sID"/>&amp;irfID=<xsl:value-of
select="$irfID"/>&amp;filename=<xsl:value-of
select="DATA_MEDIA_OBJECT/FILE/FILE_LOCATION"/>
                                        </xsl:element>
                                        <xsl:element name="MIME_TYPE">
                                              <xsl:value-of
select="DATA_MEDIA_OBJECT/FILE/MIME_TYPE"/>
                                        </xsl:element>
                                        <xsl:for-each
select="DATA_MEDIA_OBJECT/SUB_MEDIA_OBJECT">
                                              <xsl:element
name="AR_SUBOBJECT">
                                                    <xsl:element
name="URL">http://<xsl:value-of select="$host"/>:<xsl:value-of
select="$port"/>
                                                          <xsl:value-of
select="$servletPath"/>?modeID=2&amp;pID=<xsl:value-of
select="$pID"/>&amp;sID=<xsl:value-of select="$sID"/>&amp;irfID=<xsl:value-of
select="$irfID"/>&amp;filename=<xsl:value-of
select="DATA_SUB_MEDIA_OBJECT/FILE/FILE_LOCATION"/>
                                                    </xsl:element>
                                                    <xsl:element
name="MIME_TYPE">
                                                          <xsl:value-of
select="DATA_SUB_MEDIA_OBJECT/FILE/MIME_TYPE"/>
                                                    </xsl:element>
                                              </xsl:element>
                                        </xsl:for-each>
                                  </xsl:element>
                            </xsl:if>
                      </xsl:for-each>
                </xsl:if>
          </xsl:template>
          <xsl:template name="listPLITE">
                <!-- The data file is PLite -->
                <xsl:for-each select="EXHIBITION/DATA">
```

```
                    <xsl:variable name="irFile">
                          <xsl:value-of select="$contextPath"/>
                          <xsl:value-of select="FILE_LOCATION"/>
                    </xsl:variable>
                    <xsl:call-template name="listIR">
                          <xsl:with-param name="dataFile">
                                <xsl:value-of select="$irFile"/>
                          </xsl:with-param>
                    </xsl:call-template>
              </xsl:for-each>
        </xsl:template>
</xsl:stylesheet>
```

9.3.1.2 File: frp.xsl (example web style sheet)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:arco="http://www.arco-
web.org/namespaces/arco/v10">
      <xsl:output indent="yes" media-type="text/html" method="html" omit-xml-
declaration="no" doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"/>
      <xsl:param name="source"/>
      <xsl:param name="style"/>
      <!-- This parameter selects the template -->
      <xsl:param name="action"/>
      <xsl:param name="isIntro">yes</xsl:param>
      <xsl:param name="contextPath"/>
      <!-- Page number indicator on the first object. (fo % page_size) must be
equal to 1-->
      <xsl:param name="focp">1</xsl:param>
      <!-- first object of current page -->
      <xsl:param name="page_size">4</xsl:param>
      <!-- This param points to the media object to be displayed -->
      <xsl:param name="media"/>
      <!-- The session variable identifiers -->
      <xsl:param name="pID"/>
      <xsl:param name="sID"/>
      <xsl:param name="irfID"/>
      <xsl:param name="servletPath"/>
      <xsl:param name="host"/>
      <xsl:param name="port"/>
      <xsl:template match="/">
            <xsl:choose>
                  <xsl:when test="($focp mod $page_size = 1) and ($page_size
mod 2 = 0)">
                        <xsl:choose>
                              <xsl:when test="$action='main'">
                                    <xsl:call-template name="mainframe"/>
                              </xsl:when>
                              <xsl:when test="$action='nestedframe'">
                                    <xsl:call-template name="nestedframe"/>
                              </xsl:when>
                              <xsl:when test="$action='header'">
                                    <xsl:call-template name="header"/>
                              </xsl:when>
                              <xsl:when test="$action='footer'">
                                    <xsl:call-template name="footer"/>
                              </xsl:when>
                              <xsl:when test="$action='content'">
                                    <xsl:call-template name="content"/>
                              </xsl:when>
                              <!-- if you forget to specify the "action"
parameter -->
                              <xsl:otherwise>
                                    <xsl:call-template name="mainframe"/>
                              </xsl:otherwise>
                        </xsl:choose>
                  </xsl:when>
                  <xsl:otherwise>
```

```
                        <!-- Write a moderate message though :-P to explain
the error. -->
                        <h1>
                              <strong>Invalid paging system. Please check
stylesheet file.</strong>
                        </h1>
                  </xsl:otherwise>
            </xsl:choose>
      </xsl:template>
      <!-- THE MAIN FRAME -->
      <xsl:template name="mainframe">
            <html xmlns="http://www.w3.org/1999/xhtml">
                  <head>
                        <title>ARCO Virtual Exhibition</title>
                        <meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
                        <meta name="author" content="Maria Sifniotis
(M.Sifniotis@sussex.ac.uk), Anirban Basu (A.Basu@sussex.ac.uk), Nicholaos
Mourkoussis (N.Mourkoussis@sussex.ac.uk)"/>
                        <meta http-equiv="expires" content="0"/>
                  </head>
                  <frameset cols="1,*,1" frameborder="no" border="0"
framespacing="0" scrolling="no">
                        <frame src="ContentServ?modeID=3&amp;filename=s-
waste.html" name="leftFrame" id="leftFrame" frameborder="no" border="0"
framespacing="0" scrolling="no"/>
                        <frame name="mainFrame" id="mainFrame"
noresize="yes" frameborder="no" border="0" framespacing="0" scrolling="no">
                              <xsl:attribute
name="src">ContentServ?modeID=1&amp;source=<xsl:value-of
select="$source"/>&amp;action=nestedframe&amp;isIntro=<xsl:value-of
select="$isIntro"/>&amp;pID=<xsl:value-of select="$pID"/>&amp;sID=<xsl:value-
of select="$sID"/>&amp;irfID=<xsl:value-of select="$irfID"/></xsl:attribute>
                        </frame>
                        <frame src="ContentServ?modeID=3&amp;filename=s-
waste.html" name="rightFrame" id="rightFrame" frameborder="no" border="0"
framespacing="0" scrolling="no"/>
                  </frameset>
                  <noframes>
                        <h1>Sorry! Your browser is not frame-enabled.</h1>
                  </noframes>
            </html>
      </xsl:template>
      <!-- THE NESTED FRAME -->
      <xsl:template name="nestedframe">
            <html xmlns="http://www.w3.org/1999/xhtml">
                  <head>
                        <title>ARCO Virtual Exhibition</title>
                        <meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
                        <meta name="author" content="Maria Sifniotis
(M.Sifniotis@sussex.ac.uk), Anirban Basu (A.Basu@sussex.ac.uk), Nicholaos
Mourkoussis (N.Mourkoussis@sussex.ac.uk)"/>
                        <meta http-equiv="expires" content="0"/>
                  </head>
                  <frameset rows="80,*,160" frameborder="no" border="0"
framespacing="0" scrolling="no">
                        <frame name="topFrame" id="topFrame" noresize="yes"
frameborder="no" border="0" framespacing="0" scrolling="no">
                              <xsl:attribute
name="src">ContentServ?modeID=1&amp;source=<xsl:value-of
select="$source"/>&amp;action=header&amp;isIntro=<xsl:value-of
select="$isIntro"/>&amp;pID=<xsl:value-of select="$pID"/>&amp;sID=<xsl:value-
of select="$sID"/>&amp;irfID=<xsl:value-of select="$irfID"/></xsl:attribute>
                        </frame>
                        <frame name="mainFrame" id="mainFrame"
scrolling="auto" frameborder="no" border="0" framespacing="0">
                              <!-- Make the frame scroll vertically only! -
->
                              <xsl:attribute
name="src">ContentServ?modeID=1&amp;source=<xsl:value-of
```

```
select="$source"/>&amp;action=content&amp;isIntro=<xsl:value-of
select="$isIntro"/>&amp;pID=<xsl:value-of select="$pID"/>&amp;sID=<xsl:value-
of select="$sID"/>&amp;irfID=<xsl:value-of select="$irfID"/></xsl:attribute>
                          </frame>
                          <frame name="footerFrame" id="footerFrame"
noresize="yes" frameborder="no" border="0" framespacing="0" scrolling="no">
                                <xsl:attribute
name="src">ContentServ?modeID=1&amp;source=<xsl:value-of
select="$source"/>&amp;action=footer&amp;pID=<xsl:value-of
select="$pID"/>&amp;sID=<xsl:value-of select="$sID"/>&amp;irfID=<xsl:value-of
select="$irfID"/></xsl:attribute>
                          </frame>
                    </frameset>
                    <noframes>
                          <h1>Sorry! Your browser is not frame-enabled.</h1>
                    </noframes>
              </html>
        </xsl:template>
        <!-- THE HEADER FRAME -->
        <xsl:template name="header">
              <html>
                    <head>
                          <title>Header</title>
                          <meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
                          <meta name="author" content="Maria Sifniotis
(M.Sifniotis@sussex.ac.uk), Anirban Basu (A.Basu@sussex.ac.uk), Nicholaos
Mourkoussis (N.Mourkoussis@sussex.ac.uk"/>
                          <meta http-equiv="expires" content="0"/>
                          <link
href="ContentServ?modeID=3&amp;filename=css/maincss.css" rel="stylesheet"
type="text/css"/>
                          <script language="JavaScript"
src="ContentServ?modeID=3&amp;filename=js/scripts.js" type="text/javascript"/>
                    </head>
                    <body>
                          <div class="centered">
                                <table width="80%" border="0" align="center"
cellpadding="0" cellspacing="0" class="romanhaki">
                                      <tr>
                                            <td width="71">
                                                  <img
src="ContentServ?modeID=3&amp;filename=images/hhead_haki_sm.jpg" width="71"
height="74" alt=""/>
                                            </td>
                                            <td width="166" class="center">
                                                  <img
src="ContentServ?modeID=3&amp;filename=images/rom_lett_haki.gif" width="158"
height="36" alt=""/>
                                            </td>
                                            <td class="center">
                                                  <xsl:value-of
select="EXHIBITION/NAME"/>
                                            </td>
                                            <td>
                                                  <img
src="ContentServ?modeID=3&amp;filename=images/spacer.gif" width="158"
height="36" alt=""/>
                                            </td>
                                            <td width="42" class="center">
                                                  <img
src="ContentServ?modeID=3&amp;filename=images/rom_decor_haki.gif" width="38"
height="37" alt=""/>
                                            </td>
                                      </tr>
                                </table>
                          </div>
                    </body>
              </html>
        </xsl:template>
        <!-- FOOTER FRAME -->
```

```
	<xsl:template name="footer">
		<html xmlns="http://www.w3.org/1999/xhtml">
			<head>
				<title>Footer</title>
				<meta http-equiv="Content-Type" content="text/html;
charset=utf8"/>
				<meta name="author" content="Maria Sifniotis
(M.Sifniotis@sussex.ac.uk), Anirban Basu (A.Basu@sussex.ac.uk), Nicholaos
Mourkoussis (N.Mourkoussis@sussex.ac.uk"/>
				<meta http-equiv="expires" content="0"/>
				<link
href="ContentServ?modeID=3&amp;filename=css/maincss.css" rel="stylesheet"
type="text/css"/>
			</head>
			<body>
				<div class="centered">
					<table width="100%" border="0"
cellspacing="0" cellpadding="0">
						<tr>
							<td align="center"
valign="middle">
								<hr size="1"/>
							</td>
						</tr>
						<tr>
							<td align="center"
valign="middle">
								<table width="60%"
border="0" cellspacing="0" cellpadding="0">
									<tr>
										<td
align="center" valign="middle" class="bottom_links">
											<a
class="bottom_links" href="index.html" target="_top">Home</a> | <a
class="bottom_links"
href="ContentServ?modeID=4&amp;source=irf.xml&amp;style=pr_styles/irf_style.xs
l" target="_top">Today's exhibitions</a> | <a class="bottom_links"
href="#">Help</a> | <a class="bottom_links" href="http://www.arco-web.org/"
target="_top">About</a>
										</td>
									</tr>
								</table>
							</td>
						</tr>
						<tr>
							<td align="center"
valign="middle">
								<table width="60%"
border="0" cellspacing="0" cellpadding="0">
									<tr align="center"
valign="middle">
										<td
colspan="3">
											<img
src="images/spacer.gif" alt="" width="1" height="20"/>
										</td>
									</tr>
									<tr align="center"
valign="middle">
										<td>
											<a
href="http://www.arco-web.org/" target="_blank">

	<img src="ContentServ?modeID=3&amp;filename=images/Arco_logo.jpg"
alt="ARCO Consortium" width="143" height="35" border="0"/>
											</a>
										</td>
										<td>
											<a
href="http://www.cordis.lu/ist" target="_blank">
```

```
      <img src="ContentServ?modeID=3&amp;filename=images/ISTlogo.gif" alt="EU
IST" width="108" height="85" border="0"/>
                                                                      </a>
                                                                </td>
                                                                <td>
                                                                      <a
href="http://www.sussexpast.co.uk/" target="_blank">

      <img
src="ContentServ?modeID=3&amp;filename=images/sus_past_logo_pale.gif"
alt="Sussex Past" width="80" height="84" border="0"/>
                                                                      </a>
                                                                </td>
                                                          </tr>
                                                    </table>
                                              </td>
                                        </tr>
                                  </table>
                            </div>
                      </body>
                </html>
      </xsl:template>
      <!-- CONTENT FRAME -->
      <xsl:template name="content">
            <xsl:choose>
                  <xsl:when test="$isIntro='yes'">
                        <!-- Call the intro template -->
                        <xsl:call-template name="contentIntro"/>
                  </xsl:when>
                  <xsl:when test="$isIntro='no' and not($media)">
                        <xsl:call-template name="contentIR"/>
                  </xsl:when>
                  <xsl:when test="$media">
                        <xsl:call-template name="contentMedia"/>
                  </xsl:when>
            </xsl:choose>
      </xsl:template>
      <!-- INTRO FOR CONTENT FRAME -->
      <xsl:template name="contentIntro">
            <xsl:variable name="total_nodes"
select="count(EXHIBITION/DATA)"/>
            <html>
                  <head>
                        <title>Header</title>
                        <meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
                        <meta name="author" content="Maria Sifniotis
(M.Sifniotis@sussex.ac.uk), Anirban Basu (A.Basu@sussex.ac.uk), Nicholaos
Mourkoussis (N.Mourkoussis@sussex.ac.uk"/>
                        <meta http-equiv="expires" content="0"/>
                        <link
href="ContentServ?modeID=3&amp;filename=css/maincss.css" rel="stylesheet"
type="text/css"/>
                        <script language="JavaScript" type="text/javascript"
src="ContentServ?modeID=3&amp;filename=js/scripts.js"/>
                  </head>
                  <body>
                        <div class="centered">
                              <table width="80%" border="0" cellpadding="4"
cellspacing="1" id="exhibitionHolder" class="item_roman">
                                    <tr>
                                          <td colspan="2" align="left"
valign="top" class="justified_text">
                                                <!--<p
class="justified_text">
                                                      <xsl:value-of
select="EXHIBITION/DESCRIPTION"/>
                                                </p>-->
                                                <!--<p>Displaying
<xsl:value-of select="$focp"/> through
```

```
                    <xsl:if test="floor(($total_nodes - $focp) div $page_size)
&gt; 0">
                                                                    <xsl:value-of
select="$page_size + $focp - 1"/>
                                                             </xsl:if>
                                                             <xsl:if
test="floor(($total_nodes - $focp) div $page_size) &lt;= 0">
                                                                    <xsl:value-of
select="($total_nodes mod $page_size)  + $focp - 1"/>
                                                             </xsl:if> of
                    <xsl:value-of select="$total_nodes"/> total objects.</p>-->
                                                      <p>
                                                      <a>

      <xsl:attribute name="href">arif://<xsl:value-of
select="$host"/>:<xsl:value-of select="$port"/><xsl:value-of
select="$servletPath"/>?modeID=1&amp;source=<xsl:value-of
select="$source"/>&amp;action=<xsl:value-of
select="$action"/>&amp;isIntro=<xsl:value-of
select="$isIntro"/>&amp;pID=<xsl:value-of select="$pID"/>&amp;sID=<xsl:value-
of select="$sID"/>&amp;irfID=<xsl:value-of
select="$irfID"/>&amp;style=arserve.xsl&amp;arMode=listP</xsl:attribute><img
src="ContentServ?modeID=3&amp;filename=images/arlotsblack.gif" alt="List all
VRML objects of this presentation in the magic book on augmented reality
table-top" border="0"/></a>
                                                      </p>
                                                </td>
                                          </tr>
                                          <tr>
                                                <td>
                                                      <table width="70%"
border="0" cellpadding="4" cellspacing="4" class="item_roman">
                                                             <xsl:for-each
select="EXHIBITION/DATA">
                                                                    <xsl:variable
name="node_number" select="position()"/>
                                                                    <xsl:variable
name="ir_file">

      <xsl:value-of select="$contextPath"/>

      <xsl:value-of select="FILE_LOCATION"/>

      </xsl:variable>
                                                                    <xsl:variable
name="irmd_file">

      <xsl:value-of select="$contextPath"/>

      <xsl:value-of
select="document($ir_file)/INFORMATION_RESOURCE/DATA/METADATA/CODE/FILE_LOCATI
ON"/>

      </xsl:variable>
                                                                    <xsl:variable
name="comdNode"
select="document($irmd_file)/culturalObjectAMS/acquiredObject"/>
                                                                    <xsl:choose>

      <xsl:when test="($node_number &gt;= $focp) and ($node_number &lt; ($focp
+ $page_size))">

      <xsl:if test="$node_number mod 2 = 1">

      <!-- maximum 2 in a row -->

      <script type="text/javascript" language="JavaScript">
                          writeTROpenForIRIntro()
                    </script>

      </xsl:if>
```

```
	<td class="center" width="35%">

	<table border="0" cellpadding="0" cellspacing="0" class="center">

		<tr>

			<td class="center">

				<a>

					<xsl:attribute
name="href">ContentServ?modeID=1&amp;source=<xsl:value-of
select="FILE_LOCATION"/>&amp;action=content&amp;isIntro=no&amp;pID=<xsl:value-
of select="$pID"/>&amp;sID=<xsl:value-of select="$sID"/>&amp;irfID=<xsl:value-
of select="$irfID"/></xsl:attribute>

					<xsl:attribute

name="onClick">javascript:updateIsIntroHeader('ContentServ?modeID=1&amp;source
=<xsl:value-of
 select="$source"/>&amp;action=header&amp;isIntro=no&amp;pID=<xsl:value-of
select="$pID"/>&amp;sID=<xsl:value-of select="$sID"/>&amp;irfID=<xsl:value-of
select="$irfID"/>')</xsl:attribute>

					<xsl:if
test="document($ir_file)/INFORMATION_RESOURCE/DATA/THUMBNAIL">

						<img width="120" height="90"
border="0">

							<xsl:attribute
name="src">ContentServ?modeID=2&amp;filename=<xsl:value-of
select="document($ir_file)/INFORMATION_RESOURCE/DATA/THUMBNAIL/FILE_LOCATION"/
></xsl:attribute>

							<xsl:attribute
name="alt">Thumbnail for <xsl:value-of select="$comdNode/identifier"/>:
<xsl:value-of select="$comdNode/name"/></xsl:attribute>

						</img>

					</xsl:if>

					<xsl:if
test="not(document($ir_file)/INFORMATION_RESOURCE/DATA/THUMBNAIL)">

						<img
src="ContentServ?modeID=3&amp;filename=images/nothumb.gif" width="120"
height="90" border="0">

							<xsl:attribute name="alt">No
thumbnail for <xsl:value-of select="$comdNode/identifier"/>: <xsl:value-of
select="$comdNode/name"/></xsl:attribute>

						</img>

					</xsl:if>

				</a>

			</td>

		</tr>

		<tr>

			<td class="center">

				<a>
```

```
                                <xsl:attribute
name="href">ContentServ?modeID=1&amp;source=<xsl:value-of
select="FILE_LOCATION"/>&amp;action=content&amp;isIntro=no&amp;pID=<xsl:value-
of select="$pID"/>&amp;sID=<xsl:value-of select="$sID"/>&amp;irfID=<xsl:value-
of select="$irfID"/></xsl:attribute>

                                <xsl:attribute
name="onClick">javascript:updateIsIntroHeader('ContentServ?modeID=1&amp;source
=<xsl:value-of
select="$source"/>&amp;action=header&amp;isIntro=no&amp;pID=<xsl:value-of
select="$pID"/>&amp;sID=<xsl:value-of select="$sID"/>&amp;irfID=<xsl:value-of
select="$irfID"/>')</xsl:attribute>

                                <xsl:value-of select="$comdNode/name"/>

                          </a>

                    </td>

              </tr>

       </table>

       </td>

       <xsl:if test="$node_number mod 2 = 0">

       <script type="text/javascript" language="JavaScript">
                          writeTRCloseForIRIntro()
                    </script>

       </xsl:if>

       </xsl:when>
                                                                </xsl:choose>
                                                          </xsl:for-each>
                                                    </table>
                                              </td>
                                        </tr>
                                        <tr>
                                              <td>
                                                    <table width="100%">
                                                          <tr>
                                                                <td
align="right" valign="middle" width="100%">

       <xsl:if test="floor(($focp - 1) div $page_size) &gt; 0">

       <a>

       <xsl:attribute name="href">ContentServ?modeID=1&amp;source=<xsl:value-of
select="$source"/>&amp;action=content&amp;isIntro=<xsl:value-of
select="$isIntro"/>&amp;focp=<xsl:value-of select="$focp -
$page_size"/>&amp;pID=<xsl:value-of select="$pID"/>&amp;sID=<xsl:value-of
select="$sID"/>&amp;irfID=<xsl:value-of select="$irfID"/></xsl:attribute>

       <img src="ContentServ?modeID=3&amp;filename=images/left.gif" border="0">

             <xsl:attribute name="alt">Previous <xsl:value-of
select="$page_size"/> objects</xsl:attribute>

       </img>

       </a>

       </xsl:if>
                                                                </td>
                                                                <td
align="right" valign="middle">
```

```
	<xsl:if test="floor(($total_nodes - $focp) div $page_size) &gt; 0">

	<a>

	<xsl:attribute name="href">ContentServ?modeID=1&amp;source=<xsl:value-of
select="$source"/>&amp;action=content&amp;isIntro=<xsl:value-of
select="$isIntro"/>&amp;focp=<xsl:value-of select="$focp +
$page_size"/>&amp;pID=<xsl:value-of select="$pID"/>&amp;sID=<xsl:value-of
select="$sID"/>&amp;irfID=<xsl:value-of select="$irfID"/></xsl:attribute>

	<img src="ContentServ?modeID=3&amp;filename=images/right.gif"
border="0">

		<xsl:if test="($total_nodes - ($focp + $page_size)) &gt;=
$page_size">

			<xsl:attribute name="alt">Next <xsl:value-of
select="$page_size"/> objects</xsl:attribute>

		</xsl:if>

		<xsl:if test="($total_nodes - ($focp + $page_size)) &lt;
$page_size">

			<xsl:attribute name="alt">Next <xsl:value-of
select="$total_nodes mod $page_size"/> objects</xsl:attribute>

		</xsl:if>

	</img>

	</a>

	</xsl:if>
										</td>
									</tr>
								</table>
							</td>
						</tr>
					</table>
				</div>
			</body>
		</html>
	</xsl:template>
	<!-- MAIN IR FOR CONTENT FRAME -->
	<xsl:template name="contentIR">
		<xsl:variable name="irmd_file">
			<xsl:value-of select="$contextPath"/>
			<xsl:value-of
select="INFORMATION_RESOURCE/DATA/METADATA/CODE/FILE_LOCATION"/>
		</xsl:variable>
		<xsl:variable name="irmdgram_file">
			<xsl:value-of select="$contextPath"/>
			<xsl:value-of
select="INFORMATION_RESOURCE/DATA/METADATA/GRAMMAR/FILE_LOCATION"/>
		</xsl:variable>
		<xsl:variable name="comdNode"
select="document($irmd_file)/culturalObjectAMS/culturalObject"/>
		<html xmlns="http://www.w3.org/1999/xhtml">
			<head>
				<title>Information Resource - <xsl:value-of
select="INFORMATION_RESOURCE/ENTRY_ID"/>
				</title>
				<meta name="author" content="Anirban Basu, UoS
(A.Basu@sussex.ac.uk); Nicholaos Mourkoussis (N.Mourkoussis@sussex.ac.uk)"/>
				<meta http-equiv="expires" content="0"/>
				<meta http-equiv="Content-Type" content="text/html;
charset=utf-8"/>
```

```
                          <link
href="ContentServ?modeID=3&amp;filename=css/maincss.css" rel="stylesheet"
type="text/css"/>
                          <!--<script language="JavaScript"
type="text/javascript">
                                function BrowseTo(obj) {

      if(obj.options[obj.selectedIndex].value!="") {
                                              //determine the correct frame in
the frameset
                                              top.frames[1].frames[1].location
= obj.options[obj.selectedIndex].value;
                                       }
                                }
                          </script>-->
                    </head>
                    <body>
                          <div align="center">
                                <table width="70%" border="0" cellspacing="0"
cellpadding="0">
                                       <tr>
                                              <td>
                                                    <xsl:if
test="INFORMATION_RESOURCE/DATA/THUMBNAIL">
                                                          <!-- Thumbnail is
present, use it -->
                                                          <img width="160"
height="120" hspace="20" vspace="20" class="MainTableBorder">

      <xsl:attribute name="src">ContentServ?modeID=2&amp;filename=<xsl:value-
of
select="INFORMATION_RESOURCE/DATA/THUMBNAIL/FILE_LOCATION"/></xsl:attribute>

      <xsl:attribute name="alt"><xsl:value-of select="$comdNode/name"/>:
<xsl:value-of select="$comdNode/description"/></xsl:attribute>
                                                          </img>
                                                    </xsl:if>
                                                    <!-- Otherwise, display
default "no thumbnail" picture -->
                                                    <xsl:if
test="not(INFORMATION_RESOURCE/DATA/THUMBNAIL)">
                                                          <img
src="ContentServ?modeID=3&amp;filename=images/nothumb.gif" width="160"
height="120" hspace="20" vspace="20" class="MainTableBorder"/>
                                                    </xsl:if>
                                              </td>
                                              <td>
                                                    <table>
                                                          <xsl:for-each
select="$comdNode/*">
                                                                <tr>

      <xsl:variable name="xmlNode" select="current()"/>

      <xsl:for-each select="document($irmdgram_file)//xsd:element">

      <xsl:variable name="xsdNode" select="current()"/>

      <xsl:if test="$xsdNode/@name=name($xmlNode)">

      <xsl:variable name="display"
select="$xsdNode/xsd:annotation/xsd:appinfo/arco:displayName"/>

      <!-- Element exists display it -->

      <td width="25%" align="right" valign="top" class="justified_text">

             <xsl:attribute name="title"><xsl:value-of
select="$xsdNode/xsd:annotation/xsd:appinfo/arco:definition"/></xsl:attribute>

             <strong>
```

```
                    <xsl:value-of select="$display"/>

             </strong>

      </td>

      <td width="75%" align="left" valign="top" class="justified_text">

             <xsl:attribute name="title"><xsl:value-of
select="$xsdNode/xsd:annotation/xsd:appinfo/arco:content"/></xsl:attribute>

             <xsl:value-of select="$xmlNode"/>

      </td>

      </xsl:if>

      </xsl:for-each>
                                                            </tr>
                                                      </xsl:for-each>
                                                </table>
                                          </td>
                                    </tr>
                                    <xsl:for-each
select="INFORMATION_RESOURCE/DATA/MEDIA_OBJECT">
                                          <xsl:if
test="contains(DATA_MEDIA_OBJECT/FILE/MIME_TYPE,'model/vrml')">
                                                <tr>
                                                      <tr>
                                                            <td
colspan="2">
                                                                  <hr
class="item_roman" noshade="yes" width="100%" size="1"/>
                                                            </td>
                                                      </tr>
                                                      <td colspan="2">
                                                            <a>

      <xsl:attribute name="href">arif://<xsl:value-of
select="$host"/>:<xsl:value-of select="$port"/><xsl:value-of
select="$servletPath"/>?modeID=1&amp;source=<xsl:value-of
select="$source"/>&amp;action=<xsl:value-of
select="$action"/>&amp;isIntro=<xsl:value-of
select="$isIntro"/>&amp;pID=<xsl:value-of select="$pID"/>&amp;sID=<xsl:value-
of select="$sID"/>&amp;irfID=<xsl:value-of
select="$irfID"/>&amp;style=arserve.xsl&amp;arMode=selectI&amp;arSelect=<xsl:v
alue-of select="ENTRY_ID"/></xsl:attribute><img
src="ContentServ?modeID=3&amp;filename=images/aroneblack.gif" alt="Place this
VRML object on augmented reality table-top" border="0"/></a>
                                                      </td>
                                                </tr>
                                                <tr>
                                                      <td colspan="2">
                                                            <object
classid="CLSID:86A88967-7A20-11d2-8EDA-00600818EDB1"
codebase="http://www.parallelgraphics.com/bin/cortvrml.cab#Version=4,2,0,93"
width="450" height="300">
                                                                  <param
name="src">

      <xsl:attribute
name="value">ContentServ?modeID=2&amp;filename=<xsl:value-of
select="DATA_MEDIA_OBJECT/FILE/FILE_LOCATION"/></xsl:attribute>

      </param>
                                                                  <!--
<param name="mask" value="64,0,128,64,64,128,0,64"/>-->
                                                                  <param
name="vrml_background_color" value="#FFFFFF"/>
```

```
                                                                        <param
name="vrml_dashboard" value="false"/>
                                                                        <param
name="vrml_splashscreen" value="false"/>
                                                                        <param
name="contextmenu" value="true"/>
                                                                        <embed
width="450" height="300" type="model/vrml" vrml_splashscreen="false"
vrml_dashboard="false" vrml_background_color="#FFFFFF" contextmenu="true">

      <xsl:attribute name="src">ContentServ?modeID=2&amp;filename=<xsl:value-
of select="DATA_MEDIA_OBJECT/FILE/FILE_LOCATION"/></xsl:attribute>

      </embed>
                                                                  </object>
                                                            </td>
                                                      </tr>
                                                </xsl:if>
                                          </xsl:for-each>
                                    </table>
                              </div>
                        </body>
                  </html>
      </xsl:template>
</xsl:stylesheet>
```