

## XML Metadata, Schemas and XSL

### D9 Report on XML Schemas, XSL Stylesheets and X-VRML Technology

Author: PUE, UoS  
Issued by: ARCO Consortium  
Version: 1.0 Release  
Date: 25-Oct-2002

**Copyright 2002 by the ARCO Consortium:** The University of Sussex, Akademia Ekonomiczna w Poznaniu, Commissariat à L'Energie Atomique, Giunti Gruppo Editoriale, University of Bath, The Sussex Archaeological Society, and The Victoria and Albert Museum.

This document and the information contained herein may not be copied, used or disclosed in whole or in part except with prior written permission of the ARCO Consortium partners as listed above. The copyright and the foregoing restriction on copying, use disclosure extends to all media in which this information may be embodied, including magnetic storage, computer printout, visual display, etc. The document is supplied without liability for errors or omissions. Request permission to republish from: ARCO Project Co-ordinator, University of Sussex, EIT, Falmer, Brighton, BN1 9QT, UK, Tel +44 (0)1273 – 678 958 or E-mail [arco-coord@jiscmail.ac.uk](mailto:arco-coord@jiscmail.ac.uk).

## Report Documentation Page

### Report Documentation

<b>Period</b>	2
<b>Distribution</b>	Paper, Web Site, and Email
<b>Work Package</b>	WP6
<b>Deliverable</b>	D9
<b>Security</b>	Public
<b>Project Number</b>	IST-2000-28336
<b>File Name</b>	ARCO-D9-1.0-R-251002

### Report Change Log

<b>Version</b>	<b>Author(s)</b>	<b>Description</b>	<b>Date</b>
0.1, Draft	K. Walczak (PUE)	First draft, structure, X-VRML	16-Sep-2002
0.2, Draft	K. Walczak (PUE) J. Chmielewski (PUE)	ACMA AMS Manager	23-Sep-2002
0.3, Draft	K. Walczak (PUE) M. Stawniak (PUE)	X-VRML Template Manager, X-VRML Presentation Manager	25-Sep-2002
0.4, Draft	N.Mourkoussis (UoS)	AMS Implementation	27-Sep-2002
0.5, Draft	N.Mourkoussis (UoS) J. Chmielewski (PUE)	Edits in chapter 2.2.4 Table with auto generated elements	08-Oct-2002
0.6, Draft	N.Mourkoussis (UoS)	AMS Implementation AMS Schema Details	14-Oct-2002
0.7, Draft	N.Mourkoussis (UoS)	Changes to AMS schema Details due to feedback from K. Walczak (PUE) & J. Chmielewski (PUE)	17-Oct-2002
0.8, Draft	K. Walczak (PUE)	Edits in Section 2, XSL stylesheets, dynamic ARIF contents	22-Oct-2002
0.9, Final Draft	K. Walczak (PUE)	Integration, editing	24-Oct-2002
1.0, Release	N. Mourkoussis (UoS)	Release on the Web site	25-Oct-2002

Note: Insert unambiguous date, e.g. manually or using Insert > Date and Time and do not tick Update box.

## Glossary

Terms	ARCO Glossary File
For a complete glossary of ARCO terms see	ARCO-Glossary-R-1.0-280402.doc

## Summary

This document describes a suite of XML-based technologies and tools developed and used within the ARCO project. These include the XML Schemas for AMS metadata descriptions of cultural object data stored in the ARCO database, the X-VRML language used to dynamically build advanced web-based augmented reality interfaces (ARIF interfaces) of the ARCO system, and the XSL templates used to display the AMS metadata in the ARIF interfaces in a user-friendly formatted way.

This public report describes the status of these technologies developed for the second ARCO prototype, i.e. after the first year of the project. As the ARCO system will have one more prototype system followed by a final system implementation, the particular implementation elements are expected to evolve over that time. The final status of the XML technology implementation will be described in the deliverable "D11 – Final Report on XML Technology" to be published after the first two years of the ARCO project.

## Table of Contents

1.	Introduction .....	1
2.	XML Schema for AMS .....	2
2.1	Introduction .....	2
2.2	Overview of AMS Implementation .....	2
2.2.1	Structure of AMS Schema .....	2
2.2.2	Naming Convention .....	4
2.2.3	Additional Element Information .....	4
2.2.4	Auto-generated Elements .....	5
2.2.5	Application Profiles .....	7
2.3	AMS Schema Implementation.....	8
2.3.1	Cultural Object AMS.....	8
2.3.2	Acquired Object AMS.....	17
2.3.3	Refined Object AMS .....	21
2.3.4	Media Object AMS.....	23
2.3.5	Media Object Type Simple Image AMS.....	26
2.3.6	Description AMS .....	29
2.3.7	3D Studio Max Project AMS.....	31
2.3.8	VRML Model AMS .....	33
2.3.9	Panorama Image AMS .....	35
2.3.10	Multiresolution Image AMS .....	36
2.4	AMS Schema Manager .....	38
2.4.1	Cultural Object AMS Manager.....	39
2.4.2	Media Object AMS Manager.....	40
2.5	AMS Metadata Editor.....	42
3.	X-VRML Technology .....	46
3.1	Introduction to the X-VRML Language .....	46
3.2	Language Overview.....	48
3.3	Modularisation of X-VRML .....	49
3.4	X-VRML Core Module.....	49
3.4.1	Overview .....	49
3.4.2	Variables and expressions.....	49
3.4.3	Assigning values to variables .....	50
3.4.4	Inserting values .....	50
3.4.5	Conditional Statements .....	50
3.4.6	Loops.....	51
3.4.7	Nesting X-VRML Files .....	52
3.5	The X-VRML Database Module .....	53
3.5.1	Overview .....	53
3.5.2	Connecting to Databases .....	54
3.5.3	Retrieving Data from Databases .....	54
3.5.4	Updating Databases.....	55
3.6	X-VRML Object-Oriented Module.....	55
3.6.1	Concept.....	55
3.6.2	Defining X-VRML classes.....	55
3.6.3	Creating Instances of Classes.....	57
3.7	X-VRML ARCO Module .....	58
3.7.1	Overview .....	58
3.7.2	Retrieving Properties of ARIF Folders.....	58
3.7.3	Retrieving Properties of Cultural Objects.....	59
3.7.4	Retrieving Properties of Media Objects .....	60
3.7.5	Retrieving AMS data .....	60

4.	ARCO X-VRML Implementation .....	62
4.1	ARIF X-VRML Server .....	62
4.1.1	Overview .....	62
4.1.2	Architecture of the ARIF X-VRML Server .....	63
4.1.3	X-VRML Module .....	64
4.1.4	ADAM – ARCO Data Access Module .....	67
4.2	ARIF Dynamic Content .....	68
4.3	X-VRML Template Manager .....	70
4.4	X-VRML Presentation Manager .....	73
5.	XSL Stylesheets in ARCO .....	75
5.1	Overview .....	75
5.2	X-VRML – XSL Architecture .....	75
5.3	Designing XSL Templates for AMS Metadata .....	76
6.	References .....	79
	Appendix A. XML Schema for Administrative Metadata .....	80
	Appendix B. XML Schema for Cultural Object AMS .....	82
	Appendix C. XML Schema for Acquired Object AMS .....	91
	Appendix D. XML Schema for Refined Object AMS .....	96
	Appendix E. XML Schema for Media Object AMS .....	102
	Appendix F. XML Schema for Media Object Simple Image AMS .....	106
	Appendix G. XML Schema for Media Object Description AMS .....	110
	Appendix H. XML Schema for Media Object 3D Studio Max Project AMS ..	119
	Appendix I. XML Schema for Media Object VRML Model AMS .....	122
	Appendix J. XML Schema for Media Object Panorama Image AMS .....	125
	Appendix K. XML Schema for Media Object Multiresolution Image AMS...	127
	Appendix L. XML Schema for characterSet element .....	130
	Appendix M. Sample XSL Stylesheet for AMS Metadata .....	137

## List of Figures and Tables

Figure 1. The structure of AMS Schema .....	3
Figure 2. System name of the AMS element Date Created.....	4
Figure 3. Encoding of additional information in AMS XML Schemas .....	5
Table 1. List of auto-generated elements supported by ACMA .....	6
Figure 4. The architecture of application profiles in ARCO.....	7
Figure 5. The overall XML Schema of the Cultural Object AMS .....	8
Figure 6. XML Schema of the source element .....	9
Figure 7. XML Schema of the name element .....	9
Figure 8. XML Schema of the nameAlternative element .....	9
Figure 9. XML Schema of the creator element.....	10
Figure 10. XML Schema of the contributor element .....	10
Figure 11. XML Schema of the dateCreated element.....	11
Figure 12. XML Schema of the type element.....	11
Figure 13. XML Schema of the description element.....	11
Figure 14. XML Schema of the completeness element.....	12
Figure 15. XML Schema of the condition element .....	12
Figure 16. XML Schema of the productionPeriod element.....	13
Figure 17. XML Schema of the productionMethod element .....	13
Figure 18. XML Schema of the formatMedium element .....	14
Figure 19. XML Schema of the dimensions element.....	15
Figure 20. XML Schema for the coverageSpatial element.....	15
Figure 21. XML Schema of the components element.....	16
Figure 22. XML Schema of the rights element .....	16
Figure 23. XML Schema of the owner element .....	17
Figure 24. The overall XML Schema for Acquired Object AMS .....	17
Figure 25. XML Schema of the identifier element.....	18
Figure 26. XML Schema of the name element .....	18
Figure 27. XML Schema of the publisher element .....	19
Figure 28. XML Schema of the creator element .....	19
Figure 29. XML Schema of the contributor element .....	20
Figure 30. XML Schema of the dateCreated element.....	20
Figure 31. XML Schema of the description element.....	20



Figure 32.	XML Schema of the rights element.....	20
Figure 33.	XML Schema of the format element.....	21
Figure 34.	XML Schema of the formatExtent element.....	21
Figure 35.	The overall XML Schema for Refined Object AMS.....	22
Figure 36.	XML Schema of the refines element.....	23
Figure 37.	The overall XML Schema of Media Object AMS.....	23
Figure 38.	XML Schema of the name element .....	24
Figure 39.	XML Schema of the type element.....	24
Figure 40.	XML Schema of the subject element.....	25
Figure 41.	XML Schema of the description element.....	25
Figure 42.	XML Schema of the dateCreated element.....	25
Figure 43.	XML Schema of the creator element .....	26
Figure 44.	XML Schema of the formatExtent element.....	26
Figure 45.	XML Schema of the rights element.....	26
Figure 46.	The overall XML Schema for Simple Image AMS.....	27
Figure 47.	XML Schema of the technique element.....	27
Figure 48.	XML Schema of the imageSize element.....	28
Figure 49.	XML Schema of the resolution element.....	28
Figure 50.	XML Schema of the compressionMethod element .....	28
Figure 51.	XML Schema of the compressionFactor element.....	29
Figure 52.	XML Schema of the colourDepth element.....	29
Figure 53.	The overall XML Schema for Description AMS.....	30
Figure 54.	XML Schema of the technique element.....	30
Figure 55.	XML Schema of the length element.....	30
Figure 56.	The overall XML Schema for 3D Studio Max AMS.....	31
Figure 57.	XML Schema of the technique element.....	31
Figure 58.	XML Schema of the softwareVersion element .....	32
Figure 59.	XML Schema of the allVersions type .....	32
Figure 60.	XML Schema of the existingVersions type .....	32
Figure 61.	XML Schema of the requiredExtensions element.....	32
Figure 62.	The overall XML Schema for VRML Model AMS.....	33
Figure 63.	XML Schema of the technique element.....	33
Figure 64.	XML Schema of the vrmlVersion element .....	34
Figure 65.	XML Schema of the numberOfTextures element.....	34

Figure 66. XML Schema of the composite element.....	34
Figure 67. XML Schema of the animated element.....	35
Figure 68. The overall XML Schema for Panorama Image AMS.....	35
Figure 69. XML Schema of the technique element.....	35
Figure 70. XML Schema of the numberOfImages element .....	36
Figure 71. XML Schema of the stepAngle element.....	36
Figure 72. The overall XML Schema for Multiresolution Image AMS.....	36
Figure 73. XML Schema of the technique element.....	37
Figure 74. XML Schema of the resolutions element.....	37
Figure 75. XML Schema of the software element .....	38
Figure 76. XML Schema of the algorithm element .....	38
Figure 77. AMS Schema Manager – Cultural Object AMS.....	39
Figure 78. Adding new AMS schema in the ARCO AMS Schema Manager...	40
Figure 79. AMS Schema Manager – Media Object AMS.....	41
Figure 80. Creating a new AMS schema in MO AMS Manager .....	41
Figure 81. Creating a new version of AMS schema in MO AMS Manager .....	42
Figure 82. ARCO AMS Metadata Editor .....	43
Figure 83. Adding a new element in AMS Metadata Editor .....	44
Figure 84. Additional functions of the AMS Metadata Editor .....	44
Figure 85. Architecture of a conventional VRML system.....	46
Figure 86. Architecture of a model-based virtual reality system.....	48
Table 1. Comparison of Java and X-VRML versions of the same algorithm...	49
Table 2. Properties of ARIF Folders .....	59
Table 3. Properties of Cultural Objects.....	60
Table 4. Properties of Media Objects.....	60
Figure 87. ARIF X-VRML Server in the overall ARCO architecture .....	62
Figure 88. Architecture of the ARIF X-VRML Server .....	63
Figure 89. The X-VRML Module.....	64
Figure 90. Internal structure of the X-VRML Servlet.....	66
Figure 91. Retrieving binary data by the ADAM subsystem.....	67
Figure 92. ARCO Presentation Manager.....	70
Figure 93. ARCO X-VRML Template Manager .....	71
Figure 94. ARCO Template Manager – Template Parameters.....	72
Figure 95. Processing of AMS with X-VRML and XSLT .....	75

Figure 96. Designing AMS XSL template in Altova Stylesheet Designer.....76

Figure 97. ACMA XSL Manager .....77

Figure 98. Cultural Object AMS metadata formatted with XSLT .....78

## 1. Introduction

This document describes a suite of XML-based technologies and tools developed and used within the ARCO project. These include the XML Schemas for AMS metadata descriptions of cultural object data stored in the ARCO database, the X-VRML language used to dynamically build advanced web-based augmented reality interfaces (ARIF interfaces) of the ARCO system, and the XSL templates used to display the AMS metadata in the ARIF interfaces in a user-friendly formatted way.

This public report describes the status of these technologies developed for the second ARCO prototype, i.e. after the first year of the project. As the ARCO system will have one more prototype system followed by a final system implementation, the particular implementation elements are expected to evolve over that time. The final status of the XML technology implementation will be described in the deliverable "D11 – Final Report on XML Technology" to be published after the first two years of the ARCO project.

This document consists of three major parts.

The first part describes the XML implementation of the AMS metadata element sets developed within ARCO for describing the digital representations of cultural objects stored in the ARCO database. The specification of the AMS element sets is provided in the ARCO Deliverable D8 – “Report on the XML descriptions of the database cultural objects”. In this document, the overall AMS implementation rules are described followed by a detailed description of the implementation of each of the AMS element sets in XML. An overview of the set of tools designed and implemented in ARCO for management of AMS metadata schemas and AMS documents in the ARCO database is also provided.

The second part of the document describes the ARCO X-VRML technology. This includes a description of the rationale and the basics of the X-VRML language, as well as the ARCO X-VRML module, which implements the ARCO specific language extensions. Implementation of the dynamic X-VRML Server for ARIF interfaces and the set of tools for management of X-VRML templates and dynamic ARIF contents in the ARCO database is also described.

Finally, the third part of the document contains an overview of the use of XSL stylesheets in ARCO. The XSL stylesheets are used for displaying the AMS XML data in the Web ARIF interfaces, which are based on X-VRML. For this purpose, the X-VRML language has been extended in the ARCO project to support XSL processing of XML data.

## **2. XML Schema for AMS**

### **2.1 Introduction**

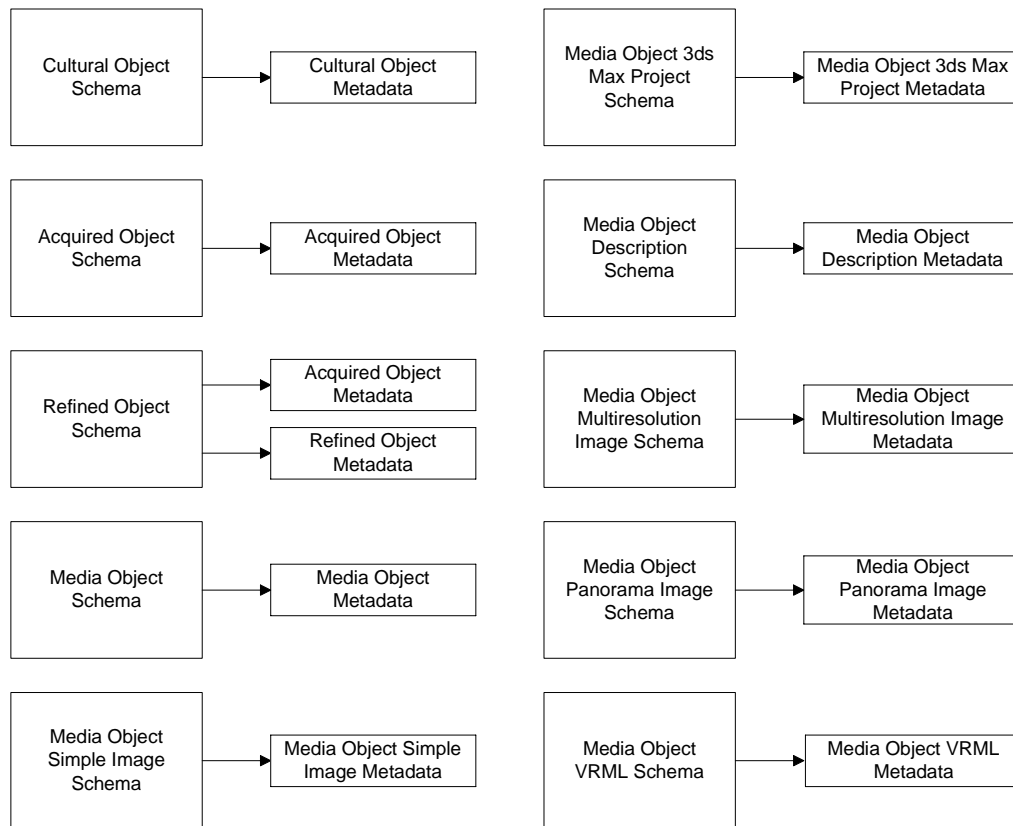
In this section, the XML implementation of the ARCO AMS metadata element sets defined in ARCO Deliverable D8 – “Report on the XML descriptions of the database cultural objects” [1] is described.

The general rules used for implementing AMS in XML are provided followed by a detailed documentation for the each implemented AMS schema. In this documentation, each AMS element is described and its simplified XML Schema is provided. The simplified schema does not contain the annotation sections, which carry additional information about AMS elements for the ARCO AMS tools. XML implementation details can be found in appendices (Appendix B. – Appendix K. ). Finally, an overview of the tools implemented in ARCO for management of AMS schemas and AMS documents in the ARCO database is provided.

### **2.2 Overview of AMS Implementation**

#### **2.2.1 Structure of AMS Schema**

The structure of AMS XML Schema is presented in Figure 1. As it can be observed from the figure, the AMS is implemented as a set of distinct schemas corresponding to the AMS element sets [1]: one for the Cultural Objects, one for the Acquired Objects, one for the Refined Objects, one including common attributes of all Media Objects, and a number of schemas with attributes specific for particular Media Object Types.



**Figure 1. The structure of AMS Schema**

As described in [1] the list Media Object Types supported by the ARCO system is extensible. A user can add new Media Object Types without a need to modify the database schema or tools. A number of pre-configured Media Object Types is installed with the ARCO system. These types are the following:

- Simple Image
- 3DS MAX project
- Description
- Multiresolution Image
- Panorama Image
- VRML model

For these Media Object Types the ARCO system provides specific AMS schemas. If custom types are added to the system, either only the general MO AMS containing attributes common to all Media Objects should be used for the new type or appropriate XML Schema file (.xsd) containing implementation of AMS elements specific to the new Media Type should be loaded into the database to extend the general MO AMS.

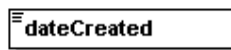
The AMS specification for the second ARCO prototype contains also a set of elements for Administrative Metadata [1] (3.1), that will be used to keep track of the creator of the metadata record and all its modifications.. Contents for these elements will be generated automatically by the ACMA application. Since the user action tracking functionality is planned for the third prototype, in this document only appropriate XML Schema implementation is provided (see Appendix A. ).

### 2.2.2 Naming Convention

The AMS element set is implemented as XML Schema and each AMS document must be a well-formed XML document. In order to create well formed XML documents the AMS element names as described in [1] had to be transformed into legal XML element names. An XML element name has the following structure [18]:

NameChar	::=	Letter   Digit   '.'   '-'   '_'   ':'   CombiningChar   Extender
Name	::=	(Letter   '_'   ':') (NameChar)*

In the implementation of ARCO AMS a uniform naming convention has been used. According to this convention all elements have names with the structure: “*lowerCaseUpperCase*”. For example, the AMS element “*Date Created*” is encoded as “*dateCreated*” in XML (see also Figure 2).



**Figure 2. System name of the AMS element Date Created**

The name used for XML element is called *system name*. The system name is used by the modules of the ARCO system for identification of the AMS elements. For user-friendly display of AMS elements in ARCO applications a different name – *display name* is used. The display name is described in the next section.

### 2.2.3 Additional Element Information

#### Display Name

Some ARCO modules must display AMS data in a user-friendly way. The system name described in the previous section is not always appropriate for this purpose. In order to enable use of more user-friendly names for AMS elements in ARCO applications, additional *display name* has been included in the AMS XML Schemas. The display name is defined in the <appinfo> sections of element annotations. In Figure 3, an example of AMS element implementation containing the <appinfo> sections is presented.

#### Unique Identifier

One of the key concepts of the ARCO project is interoperability implemented by the use of XML. This includes interoperability between ARCO components, ARCO systems as well as other systems and applications. In order to enable AMS metadata interoperability with other systems and applications each element in AMS has a unique identifier that indicates its source. The identifier is defined in the AMS XML Schema in the <appinfo> section of the element annotation (see Figure 3).

ARCO develops AMS by adopting elements from various standards. Each adopted element is encoded with its associated identifier to provide information that indicate from what standard the AMS element is adopted. Unique identifiers maximize the AMS potential for interoperability.

#### Element Definition

Another element that is included in the <appinfo> section of the element annotations is *element definition*. Its goal is to provide a statement that clearly describes the concept and essential nature of the data element. It can be used by an end-user interface in order to facilitate the user’s work (see Figure 3).

### Element Content Guideline

Another element included in the <appinfo> section of the element annotations is *content guideline*. The content guideline can be used by the end-user interface in order to provide rules for user on how to fill in the content of the element (see Figure 3).

```
<xsd:element name="name" type="xsd:string">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>MOName</arco:autoGenerated>
      <arco:displayName>Name</arco:displayName>
      <arco:identifier>http://http://purl.org/dc/elements/1.1/title</arco:identifier>
      <arco:definition>A name given to the Media Object</arco:definition>
      <arco:content>This should be a name appropriate to a particular digital
representation of the Cultural Object and should distinguish it from other
representations or manifestations
      </arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>
```

**Figure 3. Encoding of additional information in AMS XML Schemas**

### 2.2.4 Auto-generated Elements

Some of the elements that are used in the ARCO system and represented in the AMS XML Schema are generated automatically by the ARCO Content Management Application – ACMA. Values of these attributes are not entered by a user, but instead, their contents are automatically generated by the ACMA application when the AMS XML data is being written to the database.

This solution allows enforcing data integrity and releases the user from updating manually the values that can be automatically recognized and generated by the system. Examples of auto-generated elements are object size or creation date.

The following methodology was used to implement the auto-generated elements in AMS XML Schema. Each auto-generated element has an associated <arco:autoGenerated> element in the <appinfo> section of element annotation (see Figure 3).

ACMA recognizes the contents that should be generated based on the *code name* provided in the <arco:autoGenerated> element. A number of predefined code names supported by ACMA is available for schema development. The list of auto-generated elements available in the second ARCO prototype is presented in Table 1.

The auto-generated elements are displayed differently by the ACMA application allowing the user to easily recognize them (cf. Section 2.5).

Scope	Code Name	Description
Acquired Object	AO_Identifier	A prefix assigned to the museum plus the CO_ID generated by the database
Acquired Object	AO_Name	Name displayed in the objects tree
Acquired Object	AO_Description	Description displayed on AO attributes panel
Acquired Object	AO_Creator	ACMA user who created this object
Acquired Object	AO_DateCreated	Date when object was created in ACMA
Acquired Object	AO_Format	List of mime-types of all Media Objects this AO contains
Acquired Object	AO_FormatExtent	The size of all sub Media Objects



Refined Object	RO_Identifier	This will be a prefix assigned to the museum plus the CO_ID generated by the database
Refined Object	RO_Name	Same as name displayed on the objects tree
Refined Object	RO_Description	Same as description displayed on RO attributes panel
Refined Object	RO_Creator	ACMA user who created this object
Refined Object	RO_DateCreated	Date when object was created in ACMA
Refined Object	RO_Format	List of mime-types of all Media Objects this RO contains
Refined Object	RO_FormatExtent	The size of all sub Media Objects
Refined Object	RO_Refines	List of names/identifiers of Cultural Objects on the refinement path
Media Object	MO_Name	Same as name displayed on the objects tree
Media Object	MO_Type	The mime-type of this object
Media Object	MO_Creator	ACMA user who created this object
Media Object	MO_DateCreated	Date when object was created in ACMA
Media Object	MO_FormatExtent	The size of the object (in B, KB, or MB)
Simple Image	MOT_SimpleImage_ImageSize	Width and height of the image in pixels
Simple Image	MOT_SimpleImage_Resolution	Image resolution in dpi
Simple Image	MOT_SimpleImage_CompressionMethod	One of known compression methods: raw, gif, jpeg
Simple Image	MOT_SimpleImage_ColourDepth	One item from predefined directory 1,8,16,24,32,48
Description	MOT_Description_Length	Text length in chars
VRML Model	MOT_VRML_Version	One of existing versions 1.0, 2.0, 97, 200x, X3D
VRML Model	MOT_VRML_NumberOfTextures	Number of textures used by this VRML
VRML Model	MOT_VRML_Composite	Boolean that indicates whether the model is composed of more than one VRML file
VRML Model	MOT_VRML_Animated	Boolean that indicates whether the model contains animated elements
Panorama Image	MOT_PanoramaImage_NumberOfImages	Number of images in panorama view
Panorama Image	MOT_PanoramaImage_StepAngle	Step angle between images
Multiresolution Image	MOT_MultiresolutionImage_Resolutions	List of available resolutions (ex. 800x600, 1024x768, 1600x1200)

**Table 1. List of auto-generated elements supported by ACMA**

## 2.2.5 Application Profiles

The ARCO consortium investigates the concept of application profiles. An application profile is defined as the schema that consists of data elements drawn from one or more namespaces, combined by implementers, and optimised for a particular local application [5].

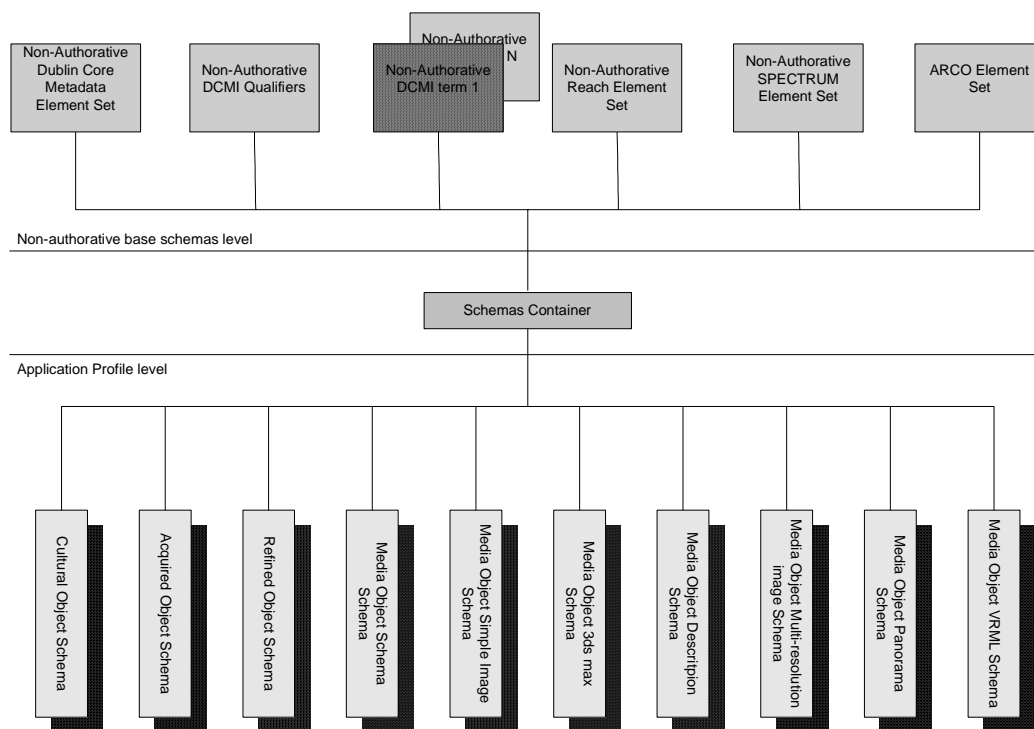
The implementation based on application profiles will be introduced in three stages, which are outlined below.

The **first** stage is mainly characterised by the XML encoding of non-authoritative version of the base schemas. Non-authoritative implies that ARCO creates its own version for each standard it adopts, by encoding only the elements used in AMS following the guidelines that the metadata body provides. Dublin Core Metadata Initiative provides guidelines [6] and proposed guidelines [7] for XML Schema implementers.

During the **second** stage, it is planned to implement an XML container schema. This schema declares XML elements to act as containers for specified elements declared in the non-authoritative versions of base schemas.

Finally, on the **third** and last stage it is planned to implement the ARCO Metadata Schemas.

Figure 4 illustrates the architecture of AMS based on application profiles.



**Figure 4. The architecture of application profiles in ARCO**

Although the concept of application profiles is elegant and suits ARCO needs, its current implementation has limitations that prevent it from being immediately used. The ARCO Consortium plans to investigate usability of application profiles in future prototypes.

## 2.3 AMS Schema Implementation

### 2.3.1 Cultural Object AMS

The Cultural Object AMS (CO AMS) is used to describe Cultural Objects stored in the ARCO Database. In this section, the XML Schema implementing the CO AMS element set defined in [1], is described. The overall CO AMS XML Schema is presented in Figure 5. The schema consists of 18 elements. These elements are described below. For actual implementation, see Appendix B.

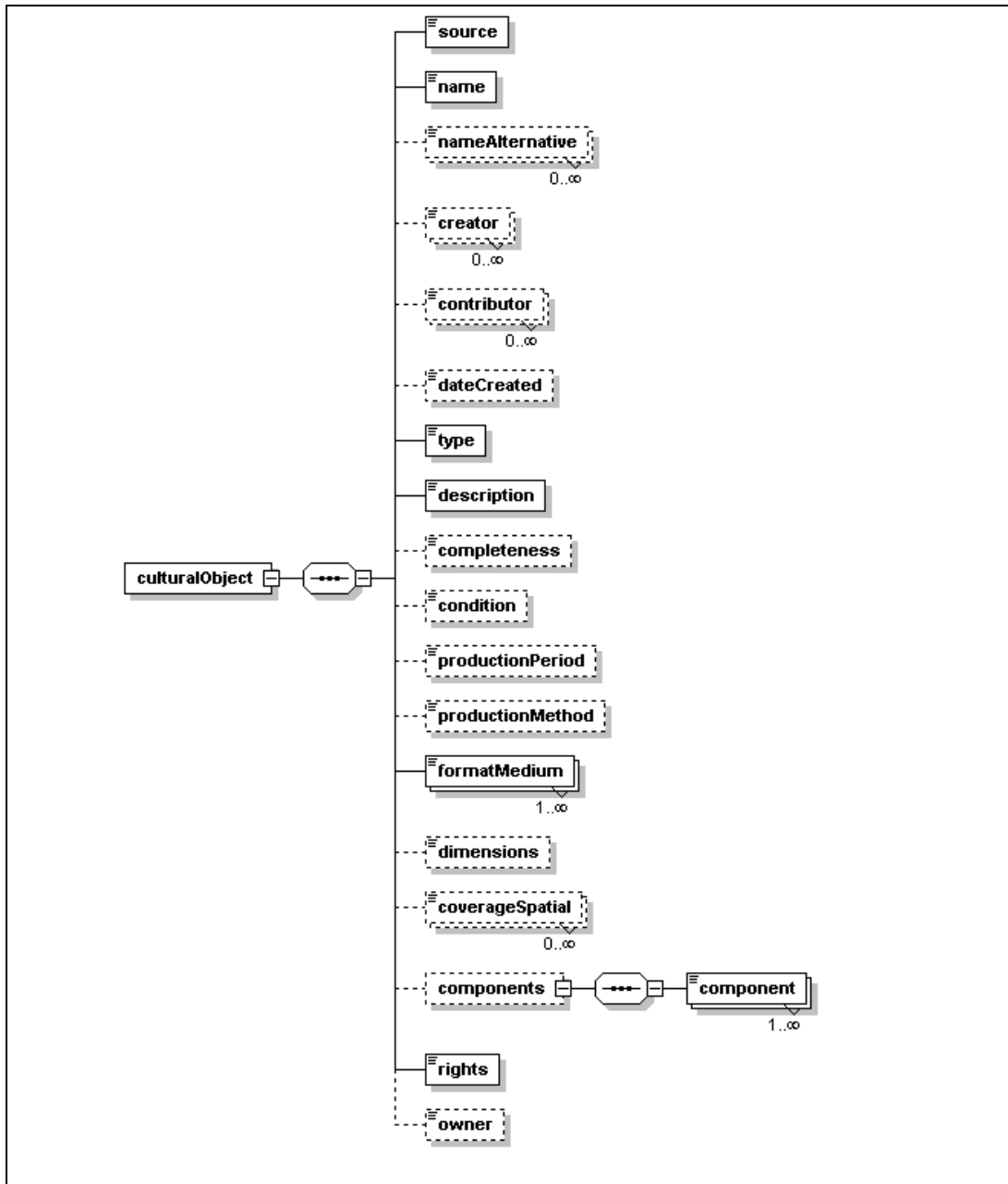


Figure 5. The overall XML Schema of the Cultural Object AMS

**Element: source (CO Source)**

The `source` element implements CO AMS element *Source* [1] (3.2.1). It provides a unique identifier for the physical artefact represented by the Cultural Object. The `source` element is a mandatory child element of the `culturalObject` element, and it must occur only once. The type of the `source` element is `xsd:string`. The possible values of the element are restricted by a regular expression. The `source` value consists of a unique museum identifier followed by the year followed by the donor identifier and the object own identifier. For example: LEWSA:1973.3.5

- LEWSA – institution
- 1973 – the year when object was acquired for the museum
- 3 – a unique number of the donor
- 5 – individual number of the object within the donor's bequest.

In Figure 6, the XML Schema for the `source` element is presented.

```
<xsd:element name="source">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\p{Lu}+:\d\d\d\d\.\d+\.\d+"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 6. XML Schema of the source element**

**Element: name (CO Name)**

The `name` element implements CO AMS element *Name* [1] (3.2.2). It contains the formal name given to the Cultural Object within the ARCO system. The `name` element is a mandatory child element of the `culturalObject` element, and it must occur only once. The type of the `name` element is `xsd:string`. In Figure 7, the XML Schema for the `name` element is presented.

```
<xsd:element name="name" type="xsd:string"/>
```

**Figure 7. XML Schema of the name element**

**Element: nameAlternative (CO Name Alternative)**

The `nameAlternative` element implements the CO AMS element *Name Alternative* [1] (3.2.3). It contains any form of the object name used as a substitute or alternative to the formal name. The `nameAlternative` element is an optional child element of the `culturalObject` element, and it may occur multiple times. The type of the element is `xsd:string` (see Figure 8).

```
<xsd:element name="nameAlternative" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
```

**Figure 8. XML Schema of the nameAlternative element**

**Element: creator (CO Creator)**

The `creator` element implements the CO AMS element *Creator* [1] (3.2.4). It describes the entity primarily responsible for the creation of the physical artefact. The `creator` element is an optional child element of the `culturalObject` element, and it may occur multiple times. The type of the element is `xsd:string`. The value of the element is restricted by either of two regular expressions. The first regular expression applies to the preferred form of entering personal names – last name first, adopting the general library indexing principles. The second regular expression is used for the organisation names. These should be entered in direct order, the larger organisation first, delimited by a comma. The XML Schema for the `creator` element is presented in Figure 9.

```
<xsd:element name="creator" minOccurs="0" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\p{Lu}\p{Ll}+, \s\p{Lu}\p{Ll}+" />
      <xsd:pattern value="\p{L}+((\s\p{L}+) | (, \s\p{L}+))*" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 9. XML Schema of the creator element**

**Element: contributor (CO Contributor)**

The `contributor` element implements the CO AMS element *Contributor* [1] (3.2.5). It describes the entity responsible for making contributions to the creation of the physical artefact. The `contributor` element is an optional child element of the `culturalObject` element, and it may occur multiple times. The type of the element is `xsd:string`. The value of the element is restricted by either of two regular expressions. The first regular expression is used for the preferred form of entering personal names – last name first, adopting the general library indexing principles. The second regular expression is used for the organisation names that should be entered in direct order, the larger organisation first, delimited by a comma. The XML Schema for the `contributor` element is presented in Figure 10.

```
<xsd:element name="contributor" minOccurs="0" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\p{Lu}\p{Ll}+, \s\p{Lu}\p{Ll}+" />
      <xsd:pattern value="\p{L}+((\s\p{L}+) | (, \s\p{L}+))*" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 10. XML Schema of the contributor element**

### Element: dateCreated (CO Date Created)

The `dateCreated` element implements the CO AMS element *Date Created* [1] (3.2.6). It represents the date of creation of the physical artefact. The `dateCreated` element is an optional child element of the `culturalObject` element, and it may occur only once. The type of the element is `xsd:string`. The value of the element is restricted by a regular expression. The regular expression limits the possible values to one the following formats:

- YYYY (year)
- YYYY-MM (year-month)
- YYYY-MM-DD (year-month-day)
- YYYY=YYYY (year=year)
- period description in a textual form

The XML Schema for the `dateCreated` element is presented in Figure 11.

```
<xsd:element name="dateCreated" minOccurs="0">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="(\d\d\d\d)|(\d\d\d-\d\d\d\d)|(\d\d\d-\d\d\d-\d\d\d\d)|(\d\d\d\d=\d\d\d\d)|(\p{L}+(\s\p{L}+)|(,\s\p{L}+))*"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Figure 11. XML Schema of the `dateCreated` element

### Element: type (CO Type)

The `type` element implements the CO AMS element *Type* [1] (3.2.7). It defines the nature or genre of the cultural object. The `type` element is a mandatory child element of the `culturalObject` element and it must occur only once. The type of the element is `xsd:string` (see Figure 12).

```
<xsd:element name="type" type="xsd:string"/>
```

Figure 12. XML Schema of the `type` element

### Element: description (CO Description)

The `description` element implements the CO AMS element *Description* [1] (3.2.8). It contains a short description characterising the physical artefact. The `description` element is mandatory child element of the `culturalObject` element, and it must occur only once. The type of the element is `xsd:string` (see Figure 13).

```
<xsd:element name="description" type="xsd:string"/>
```

Figure 13. XML Schema of the `description` element

**Element: completeness (CO Completeness)**

The `completeness` element implements the CO AMS element *Completeness* [1] (3.2.9). It defines the completeness of the physical artefact. The `completeness` element is optional child element of the `culturalObject` element, and it may occur only once. The type of the element is `xsd:string`. The element is restricted to accept only values from a predefined list: Complete, Incomplete, or Uncertain. The XML Schema for the `completeness` element is presented in Figure 14.

```
<xsd:element name="completeness" minOccurs="0">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Complete"/>
      <xsd:enumeration value="Incomplete"/>
      <xsd:enumeration value="Uncertain"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 14. XML Schema of the completeness element**

**Element: condition (CO Condition)**

The `condition` element implements the CO AMS element *Condition* [1] (3.2.10). It describes the condition of the physical artefact. The `condition` element is optional child element of the `culturalObject` element, and it may occur only once. The type of the element is `xsd:string`. The element values are restricted to a predefined list: Poor, Fair, and Good (see Figure 15).

```
<xsd:element name="condition" minOccurs="0">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Poor"/>
      <xsd:enumeration value="Fair"/>
      <xsd:enumeration value="Good"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 15. XML Schema of the condition element**

**Element: productionPeriod (CO Production Period)**

The `productionPeriod` element implements the CO AMS element *Production Period* [1] (3.2.11). It contains 'the date when a stage in the design, creation or manufacture of the object took place', which allows historical periods to be expressed. The `productionPeriod` is an

optional child element of the `culturalObject` element, and it may occur only once. The type of the element is `xsd:string` restricted by a regular expression to accept values either in textual form, or numerical form e.g., 1830=1860 (the = sign is always used to indicate a period of time). The XML Schema of the `productionPeriod` element is presented in Figure 16.

```
<xsd:element name="productionPeriod" minOccurs="0">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="(\d\d\d\d\d\d\d\d)|(\p{L}+((\s\p{L}+)|(\s\p{L}+))*")"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 16. XML Schema of the `productionPeriod` element**

**Element: `productionMethod` (CO Production Method)**

The `productionMethod` element implements the CO AMS element *Production Method* [1] (3.2.12). It describes the processes, methods, techniques and tools used to fabricate or decorate the object. The `productionMethod` is an optional child element of the `culturalObject` element, and it may occur only once. The type of the element is `xsd:string` (see Figure 17).

```
<xsd:element name="productionMethod" type="xsd:string" minOccurs="0"/>
```

**Figure 17. XML Schema of the `productionMethod` element**

**Element: `formatMedium` (CO Format Medium)**

The `formatMedium` element implements the CO AMS element *Format Medium* [1] (3.2.13). It describes the substance(s) of which the object is made. The `formatMedium` element is a mandatory child element of the `culturalObject` element, and it may occur multiple times. The type of the element is `xsd:string`. The element values are restricted to a predefined list: *Amber, Tin, Brass, Wood Steel Metal (bell), Tin (plated), Copper, Iron (wrought), Boxwood, Mahogany, Brass (cast), Lead, Bronze, Maple, Ivory, Iron, and Glass* (see Figure 18).



```
<xsd:element name="formatMedium" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Amber"/>
      <xsd:enumeration value="Boxwood"/>
      <xsd:enumeration value="Brass"/>
      <xsd:enumeration value="Brass (cast)"/>
      <xsd:enumeration value="Bronze"/>
      <xsd:enumeration value="Copper"/>
      <xsd:enumeration value="Glass"/>
      <xsd:enumeration value="Iron"/>
      <xsd:enumeration value="Iron (wrought)"/>
      <xsd:enumeration value="Ivory"/>
      <xsd:enumeration value="Lead"/>
      <xsd:enumeration value="Mahogany"/>
      <xsd:enumeration value="Maple"/>
      <xsd:enumeration value="Metal (bell)"/>
      <xsd:enumeration value="Steel"/>
      <xsd:enumeration value="Tin"/>
      <xsd:enumeration value="Tin (plated)"/>
      <xsd:enumeration value="Wood"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 18. XML Schema of the formatMedium element**

#### **Element: dimensions (CO Dimensions)**

The dimensions element implements the CO AMS element *Dimensions* [1] (3.2.14). It provides the measurements associated with the three dimensions of the object: height, width and depth. The dimensions element is an optional child element of the `culturalObject` element, and it may occur only once. The type of the element is `xsd:string`. The element value is restricted by a regular expression which enforces the correct format: *heightxwidthxdepth* (see Figure 19).

```
<xsd:element name="dimensions" minOccurs="0">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d+\sx\s\d+\sx\s\d+"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 19. XML Schema of the dimensions element**

**Element: coverageSpatial (CO Coverage Spatial)**

The coverageSpatial element implements the CO AMS element *Coverage Spatial* [1] (3.2.15). It provides the geographical location in which an object was created. The coverageSpatial element is an optional child element of the culturalObject element, and it may occur multiple times. The type of the element is xsd:string. The element value is restricted by a regular expression to enforce a format in a coma-separated list of strings (see Figure 20).

```
<xsd:element name="coverageSpatial" minOccurs="0" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 20. XML Schema for the coverageSpatial element**

**Element: components (CO Components)**

The components element implements the CO AMS element *Components* [1] (3.2.16). It contains description of any parts/pieces of the work of art. The components element is an optional child element of the culturalObject element, and it may occur only once. The type of the element is defined as a sequence of xsd:string elements. This element should be used when the physical artefact consists of more than one component (see Figure 21).

```
<xsd:element name="components" minOccurs="0">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="component" type="xsd:string" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

**Figure 21. XML Schema of the components element**

**Element: rights (CO Rights)**

The `rights` element implements the CO AMS element *Rights* [1] (3.2.17). It contains information about rights held in and over the cultural artefact. The `rights` element is a mandatory child element of the `culturalObject` element, and it may occur only once. The type of the element is `xsd:string` (see Figure 22).

```
<xsd:element name="rights">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>
</xsd:element>
```

**Figure 22. XML Schema of the rights element**

**Element: owner (CO Owner)**

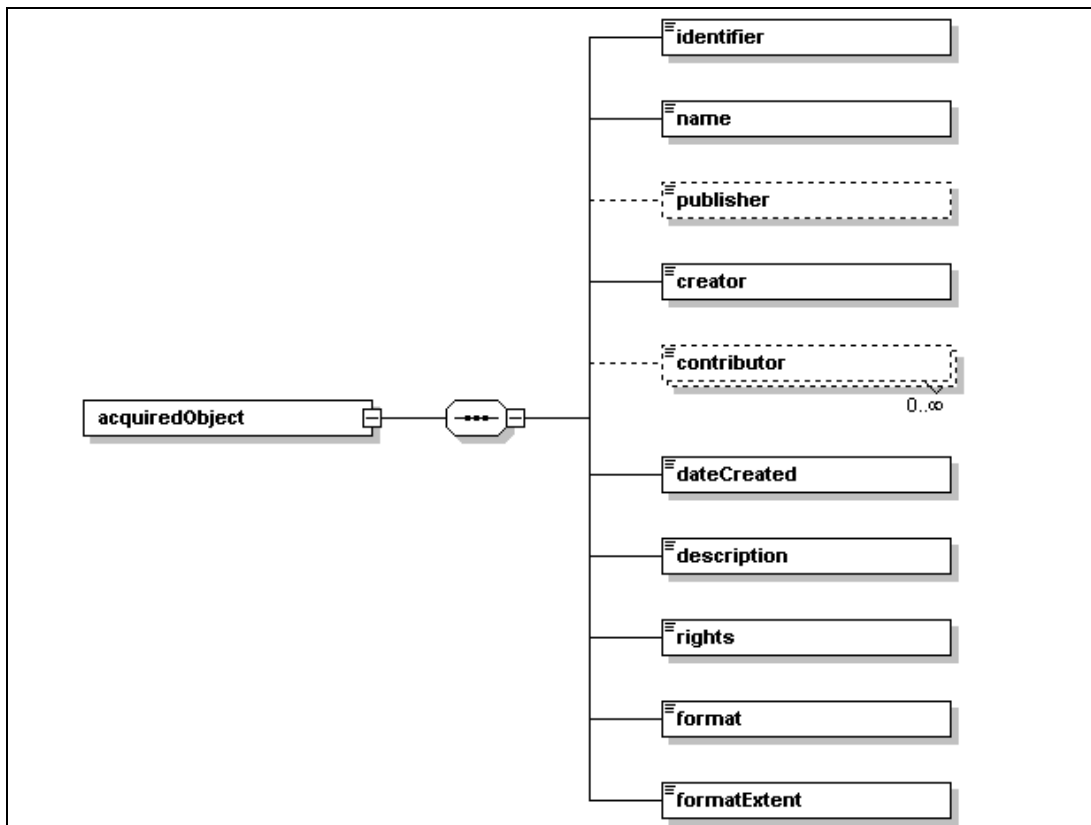
The `owner` element implements the CO AMS element *Owner* [1] (3.2.18). It provides the name of the institution or individual who owns the artefact. The `owner` element is an optional child element of the `culturalObject` element, and it may occur only once. The type of the element is `xsd:string`. The value of the element is restricted by either of two regular expressions. The first regular expression defines the preferred form to enter personal names – last name first, adopting the general library indexing principles. The second regular expression defines that the organisation names should be entered in direct order, the larger organisation first, delimited by a comma (see Figure 23).

```
<xsd:element name="owner" minOccurs="0">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\p{Lu}\p{Ll}+, \s\p{Lu}\p{Ll}+ " />
      <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 23. XML Schema of the owner element**

### 2.3.2 Acquired Object AMS

The Acquired Object AMS (AO AMS) is used to describe Acquired Objects stored in the ARCO Database. In this section the XML Schema implementing of the AO AMS element set defined in [1] is described. The overall AO AMS schema is presented in Figure 24. The schema consists of 10 elements. These elements are described below. For actual implementation see Appendix C.



**Figure 24. The overall XML Schema for Acquired Object AMS**

#### **Element: identifier (AO Identifier)**

The `identifier` element implements the AO AMS element *Identifier* [1] (3.3.1). It provides a unique identifier assigned to the Acquired Object a particular digital representation of the cultural object. The `identifier` element is a mandatory child element of the `acquiredObject` element, and it must occur only once. The type of the element is `xsd:string` restricted by a regular expression, which defines that a valid identifier value is a prefix assigned to the museum (string) concatenated with the ID generated by the database (number). The element is auto-generated by the use of the `AO_Identifier` code name. The XML Schema for the `identifier` element is presented in Figure 25.

```
<xsd:element name="identifier">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\p{Lu}+\d+"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 25. XML Schema of the identifier element**

#### **Element: name (AO Name)**

The `name` element implements the AO AMS element *Name* [1] (3.3.2). It provides the name of the acquired object – a digital manifestation of an artefact. The `name` element is a mandatory child element of the `acquiredObject` element, and it must occur only once. The type of the element is `xsd:string` (see Figure 26). The element is auto-generated by the use of the `AO_Name` code name.

```
<xsd:element name="name" type="xsd:string"/>
```

**Figure 26. XML Schema of the name element**

#### **Element: publisher (AO Publisher)**

The `publisher` element implements the AO AMS element *Publisher* [1] (3.3.3). It contains information about entity responsible for making the resource available or accessible to others. The `publisher` element is an optional child element of the `acquiredObject` element, and it may occur only once. The type of the element is `xsd:string`. The value of the element is restricted by either of two regular expressions. The first regular expression applies to the preferred form of entering personal names – last name first, adopting the general library indexing principles. The second regular expression is used for the organisation names. These should be entered in direct order, the larger organisation first, delimited by a comma (see Figure 27).

```
<xsd:element name="publisher" minOccurs="0">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*"/>
      <xsd:pattern value="\p{Lu}\p{Ll}+,\s\p{Lu}\p{Ll}+"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 27. XML Schema of the publisher element**

**Element: creator (AO Creator)**

The `creator` element implements the AO AMS element *Creator* [1] (3.3.4). It contains information about an entity primarily responsible for the creation of the digital manifestation of the artefact. The `creator` element is a mandatory child element of the `acquiredObject` element, and it must occur only once. The type of the element is `xsd:string` restricted by a regular expression defining the preferred format of entering personal names – last name first, adopting the general library indexing principles (see Figure 28). The element is auto-generated by ACMA with the `AO_Creator` code name. The values are constrained to the list of registered and privileged ARCO users.

```
<xsd:element name="creator">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 28. XML Schema of the creator element**

**Element: contributor (AO Contributor)**

The `contributor` element implements the AO AMS element *Contributor* [1] (3.3.5). It contains information about an entity contributing to the creation of the digital manifestation of the cultural object. The `contributor` element is an optional child element of the `acquiredObject` element, and it may occur multiple times. The type of the element is `xsd:string`. The value of the element is restricted by either of two regular expressions. The value of the element is restricted by either of two regular expressions. The first regular expression applies to the preferred form of entering personal names – last name first, adopting the general library indexing principles. The second regular expression is used for the organisation names. These should be entered in direct order, the larger organisation first, delimited by a comma (see Figure 29).

```
<xsd:element name="contributor" minOccurs="0" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\p{Lu}\p{Ll}+,\s\p{Lu}\p{Ll}+"/>
      <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 29. XML Schema of the contributor element**

**Element: dateCreated (AO Date Created)**

The dateCreated element implements the AO AMS element *Date Created* [1] (3.3.6). It defines the date of creation of the Acquired Object. The dateCreated element is a mandatory child element of the acquiredObject element, and it must occur only once. The type of the element is xsd:dateTime. The values are constrained to be of W3C Encoding rules for dates and times, where YYYY-MM-DDThh:mm:ssTZD is the chosen format. The element is auto-generated by ACMA with the AO\_DateCreated code name. The XML Schema for the dateCreated element is presented in Figure 30.

```
<xsd:element name="dateCreated" type="xsd:dateTime"/>
```

**Figure 30. XML Schema of the dateCreated element**

**Element: description (AO Description)**

The description element implements the AO AMS element *Description* [1] (3.3.7). It contains a short description characterising the digital representation of the artefact. The description element is a mandatory child element of the acquiredObject element, and it must occur only once. The type of the element is xsd:string. The element is auto-generated by ACMA with the AO\_Description code name. The XML Schema for the description element is presented in Figure 31.

```
<xsd:element name="description" type="xsd:string"/>
```

**Figure 31. XML Schema of the description element**

**Element: rights (AO Rights)**

The rights element implements the AO AMS element *Rights* [1] (3.3.8). It contains information about rights held in and over the Acquired Object. The rights element is a mandatory child element of the acquiredObject element, and it must occur only once. The type of the element is xsd:string (see Figure 32).

```
<xsd:element name="rights" type="xsd:string"/>
```

**Figure 32. XML Schema of the rights element**

**Element: format (AO Format)**

The `format` element implements the AO AMS element *Format* [1] (3.3.9). It describes the available formats of the Acquired Object. The `format` element is a mandatory child element of the `acquiredObject` element, and it must occur only once. The type of the element is a list of `xsd:string` elements with a regular expression constraining possible list element values to a standard MIME Type format (see Figure 33). The `format` element is auto-generated by ACMA with the `AO_Format` code name.

```
<xsd:element name="format">
  <xsd:simpleType>
    <xsd:list>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="\P{Ll}(\P{L}|\d|\+|\-|/|\.)+" />
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:list>
  </xsd:simpleType>
</xsd:element>
```

**Figure 33. XML Schema of the `format` element**

**Element: formatExtent (AO Format Extent)**

The `formatExtent` element implements the AO AMS element *Format Extent* [1] (3.3.10). It defines the digital size of the Acquired Object. The `formatExtent` element is a mandatory child element of the `acquiredObject` element, and it must occur only once. The type of the element is based on `xsd:string` with a regular expression enforcing the correct format as a number followed by white space and the unit code - B, KB or MB (see Figure 34). The `formatExtent` element is auto-generated by ACMA with the `AO_FormatExtent` code name.

```
<xsd:element name="formatExtent">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d+\s(B|KB|MB)" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 34. XML Schema of the `formatExtent` element**

### 2.3.3 Refined Object AMS

The refined Object AMS (RO AMS) is used to describe the Refined Objects stored in the ARCO Database. The RO AMS for the second prototype of the ARCO system is the same as



AO AMS with addition of one element. The implementation of elements originating from the AO AMS is the same as in AO AMS with the differences only in the annotation sections (code names for auto-generated elements, display names, element definitions, and content guidelines) – for details see Appendix D.

The overall RO AMS schema is presented in Figure 35. The schema consists of 11 elements. The `identifier`, `name`, `publisher`, `creator`, `contributor`, `dateCreated`, `description`, `rights`, `format`, and `formatExtent` are the same as in the AO AMS XML Schema. One new element: `refines` is described below.

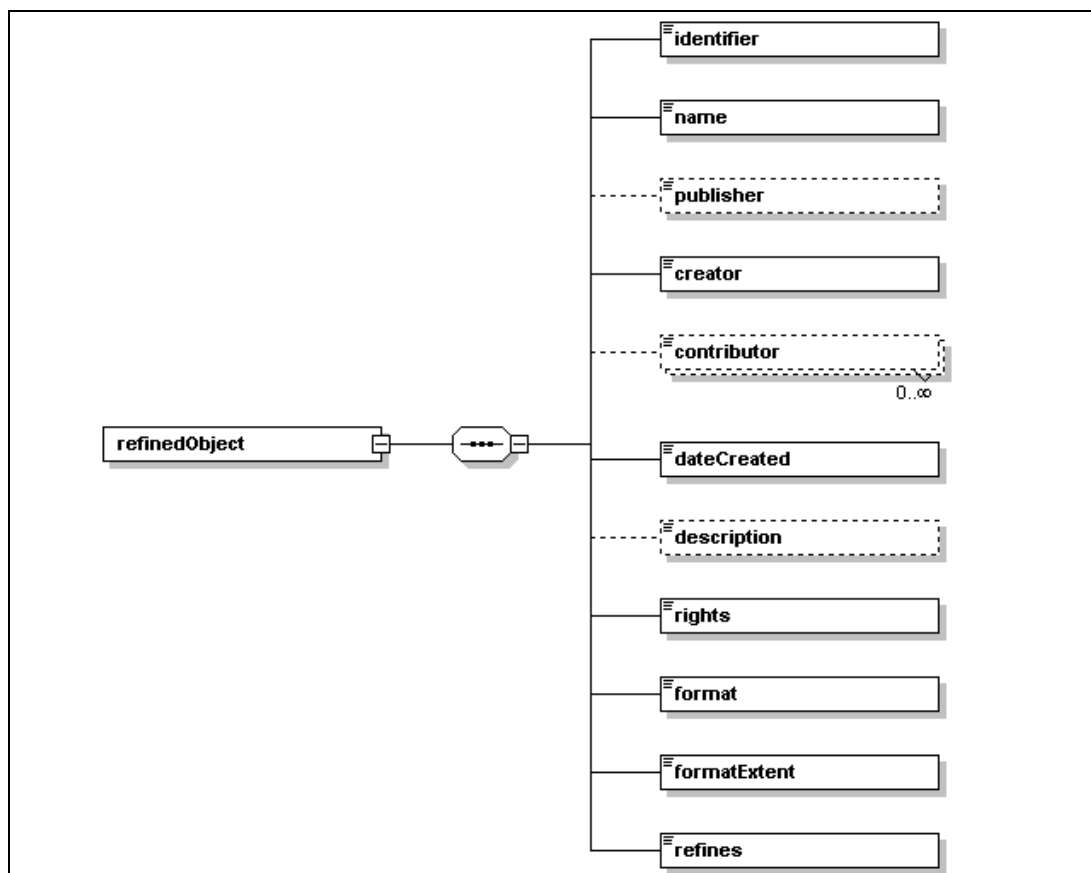


Figure 35. The overall XML Schema for Refined Object AMS

#### Element: `refines` (RO Refines)

The `refines` element implements the RO AMS element *Refines* [1] (3.4.1). It provides an ordered list of identifiers of Cultural Objects (Acquired Objects and Refined Objects) on the refinement path. The type of the `refines` element is a list of `xsd:string` elements with a regular expression restricting the format of list elements to a string followed by a number (see AO Identifier in Section 2.3.2). The XML Schema for the `refines` element is presented in Figure 36. The `refines` element is auto-generated by ACMA with the `RO_Refines` code name.

```
<xsd:element name="refines">
  <xsd:simpleType>
    <xsd:list>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="\p{Lu}+\d+"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:list>
  </xsd:simpleType>
</xsd:element>
```

Figure 36. XML Schema of the refines element

#### 2.3.4 Media Object AMS

The Media Object AMS (MO AMS) is used to describe the Media Objects stored in the ARCO Database. In this section the XML Schema, implementing of the MO AMS element set defined in [1], is described. The overall MO AMS schema is presented in Figure 37. The schema consists of eight elements. These elements are described below. For actual implementation see Appendix E.

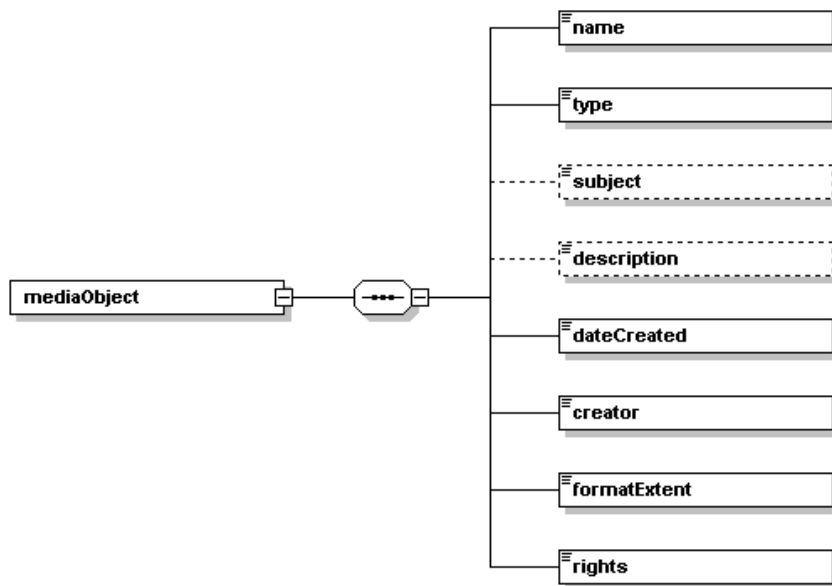


Figure 37. The overall XML Schema of Media Object AMS

##### Element: name (MO Name)

The name element implements the MO AMS element *Name* [1] (3.5.1). It contains the name of the Media Object. The name element is a mandatory child element of the mediaObject

element, and it must occur only once. The type of the element is `xsd:string` (see Figure 38). The name element is auto-generated by ACMA with the code name `MO_Name`.

```
<xsd:element name="name" type="xsd:string"/>
```

**Figure 38. XML Schema of the name element**

#### **Element: type (MO Type)**

The `type` element implements the MO AMS element *Type* [1] (3.5.2). It defines the type of the Media Object. The `type` element is a mandatory child element of the `mediaObject` element, and it must occur only once. The type of the element is a list of `xsd:string` elements restricted with a regular expression that allows to create a valid MIME Type name (see Figure 39). The `type` element is auto-generated by ACMA with the code name `MO_Type`.

```
<xsd:element name="type">
  <xsd:simpleType>
    <xsd:list>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="\P{Ll}(\P{L}|\d|\+|\-|/|\.)+"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:list>
  </xsd:simpleType>
</xsd:element>
```

**Figure 39. XML Schema of the type element**

#### **Element: subject (MO Subject)**

The `subject` element implements the MO AMS element *Subject* [1] (3.5.3). The *subject* element contains the keywords that identify the subjects of the contents of the Media Object. The `subject` element is an optional child element of the `mediaObject` element, and it must occur only once. The type of the element is a list of `xsd:string` elements with a regular expression enforcing the keywords to contain a upper-case letter followed by a number of lower-case letters (see Figure 40).

```
<xsd:element name="subject">
  <xsd:simpleType>
    <xsd:list>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="\P{Lu}\P{Ll}+"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:list>
  </xsd:simpleType>
</xsd:element>
```

**Figure 40. XML Schema of the subject element**

**Element: description (MO Description)**

The description element implements the MO AMS element *Description* [1] (3.5.4). It contains a short description characterising the Media Object. The description element is an optional child element of the mediaObject element, and it may occur only once. The type of the element is xsd:string (see Figure 41).

```
<xsd:element name="description" type="xsd:string" minOccurs="0"/>
```

**Figure 41. XML Schema of the description element**

**Element: dateCreated (MO Date Created)**

The dateCreated element implements the MO AMS element *Date Created* [1] (3.5.5). It contains the date of creation of the Media Object. The dateCreated element is a mandatory child element of the mediaObject element, and it must occur only once. The type of the element is xsd:dateTime. The values must follow W3C Encoding rules for dates and times, where YYYY-MM-DDThh:mm:ssTZD is the selected format (see Figure 42). The dateCreated element is auto-generated by ACMA with the code name MO\_DateCreated.

```
<xsd:element name="dateCreated" type="xsd:dateTime"/>
```

**Figure 42. XML Schema of the dateCreated element**

**Element: creator (MO Creator)**

The creator element implements the MO AMS element *Creator* [1] (3.5.6). It contains information about an entity primarily responsible for the creation of the Media Object. The creator element is a mandatory child element of the mediaObject element, and it may occur only once. The type of the element is a list of xsd:string elements constrained by a regular expression defining the preferred format of entering personal names – last name first, adopting the general library indexing principles (see Figure 43). The element is auto-generated by ACMA with the MO\_Creator code name. The values are constrained to the list of registered and privileged ARCO users.

```
<xsd:element name="creator">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\P{Lu}\P{Ll}+(\, \s\P{Lu}\P{Ll}+)*"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 43. XML Schema of the creator element**

**Element: formatExtent (MO Format Extent)**

The `formatExtent` element implements the RO AMS element *Format Extent* [1] (3.5.7). It provides the digital size of the Media Object. The `formatExtent` element is a mandatory child element of the `refinedObject` element, and it must occur only once. The type of the element is based on `xsd:string` with a regular expression enforcing the correct format as a number followed by white space and the unit code - B, KB or MB (see Figure 44). The `formatExtent` element is auto-generated by ACMA with the `MO_FormatExtent` code name.

```
<xsd:element name="formatExtent">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d+\s(B|KB|MB)"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 44. XML Schema of the formatExtent element**

**Element: rights (MO Rights)**

The `rights` element implements the RO AMS element *Rights* [1] (3.5.8). It contains information about rights held in and over the Media Object. The `rights` element is a mandatory child element of the `mediaObject` element, and it must occur only once. The type of the element is `xsd:string` (see Figure 45).

```
<xsd:element name="rights" type="xsd:string"/>
```

**Figure 45. XML Schema of the rights element**

### 2.3.5 Media Object Type Simple Image AMS

The Media Object Type Simple Image AMS (MOT Simple Image) is used to describe the Media Objects of type Simple Image stored in the ARCO Database. In this section, the XML Schema implementing of the MOT Simple Image AMS element set defined in [1], is described. The overall MOT Simple Image AMS schema is presented in Figure 46. The schema consists of six elements. These elements are described below. For actual implementation see Appendix F.

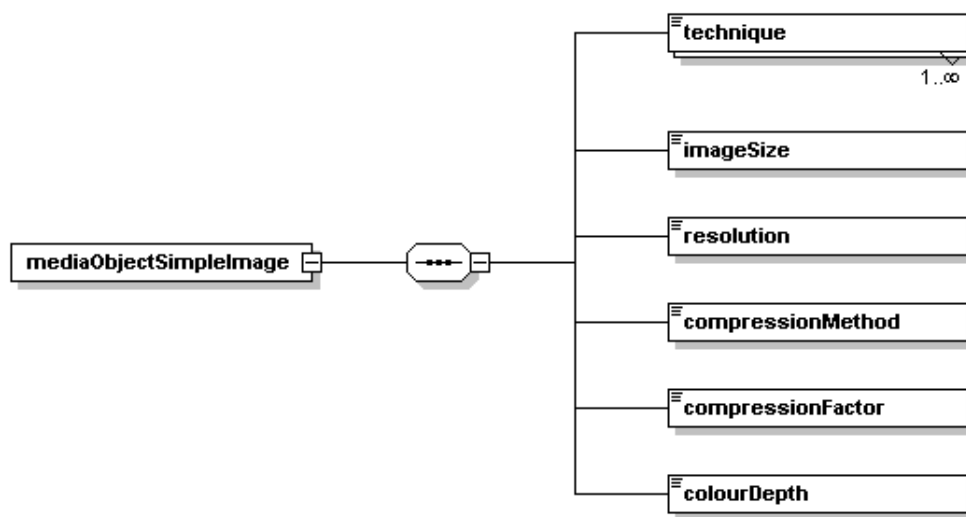


Figure 46. The overall XML Schema for Simple Image AMS

#### Element: technique (MOT Simple Image: Technique)

The *technique* element implements the MOT Simple Image AMS element *Technique* [1] (3.6.1.1). It contains information about the technique used to acquire Media Object. The *technique* element is a mandatory child element of the *mediaObjectSimpleImage* element, and may occur multiple times. The type of the element is `xsd:string`. The value of the element can be only one from the predefined list: *Digital Photography*, *Scanning*, and *Manual Painting* (see Figure 47).

```

<xsd:element name="technique" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Digital Photography"/>
      <xsd:enumeration value="Manual Painting"/>
      <xsd:enumeration value="Scanning"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
  
```

Figure 47. XML Schema of the technique element

#### Element: imageSize (MOT Simple Image: Image Size)

The *imageSize* element implements the MOT Simple Image AMS element *Image Size* [1] (3.6.1.2). It provides information about the dimensions of the image. The *imageSize* element is a mandatory child element of the *mediaObjectSimpleImage* element, and it must occur only once. The type of the element is `xsd:string` restricted by a regular expression to allow values in format *numberxnumber* (see Figure 48). The *imageSize* element is auto-generated by ACMA with the code name `MOT_SimpleImage_ImageSize`.

```
<xsd:element name="imageSize">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d+x\d+"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 48. XML Schema of the imageSize element**

**Element: resolution (MOT Simple Image: Resolution)**

The resolution element implements the MOT Simple Image AMS element *Resolution* [1] (3.6.1.3). It provides the image resolution in dpi. The resolution element is a mandatory child element of the mediaObjectSimpleImage element, and it must occur only once. The type of the element is `xsd:string` (see Figure 49). The resolution element is auto-generated by ACMA with the code name `MOT_SimpleImage_Resolution`.

```
<xsd:element name="resolution" type="xsd:string"/>
```

**Figure 49. XML Schema of the resolution element**

**Element: compressionMethod (MOT Simple Image: Compression Method)**

The compressionMethod element implements the MOT Simple Image AMS element *Compression method* [1] (3.6.1.4). It contains information about the compression method used to create the image file. The compressionMethod element is a mandatory child element of the mediaObjectSimpleImage element, and it must occur only once. The type of the element is `xsd:string`. The value of the element can be only one from the predefined list: *raw*, *jpeg*, *gif* (see Figure 50). The element is auto-generated by ACMA with the code name `MOT_SimpleImage_CompressionMethod`.

```
<xsd:element name="compressionMethod">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="raw"/>
      <xsd:enumeration value="jpeg"/>
      <xsd:enumeration value="gif"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 50. XML Schema of the compressionMethod element**

**Element: compressionFactor (MOT Simple Image: Compression Factor)**

The `compressionFactor` element implements the MOT Simple Image AMS element *Compression factor* [1] (3.6.1.5). It provides the image compression algorithm strength factor. The `compressionFactor` element is a mandatory child element of the `mediaObjectSimpleImage` element, and it must occur only once. The type of the element is `xsd:string`. The value of the element must be one from the predefined list: *High*, *Low*, *Medium*, *None* (see Figure 51).

```
<xsd:element name="compressionFactor">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="High"/>
      <xsd:enumeration value="Low"/>
      <xsd:enumeration value="Medium"/>
      <xsd:enumeration value="None"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 51. XML Schema of the `compressionFactor` element**

**Element: colourDepth (MOT Simple Image: Colour Depth)**

The `colourDepth` element implements the MOT Simple Image AMS element *Colour depth* [1] (3.6.1.6). It contains the number of bits that represents single colour point. The `colourDepth` element is a mandatory child element of the `mediaObjectSimpleImage` element, and it must occur only once. The type of the element is `xsd:integer` (see Figure 52). The element is auto-generated by ACMA with the code name `MOT_SimpleImage_ColourDepth`.

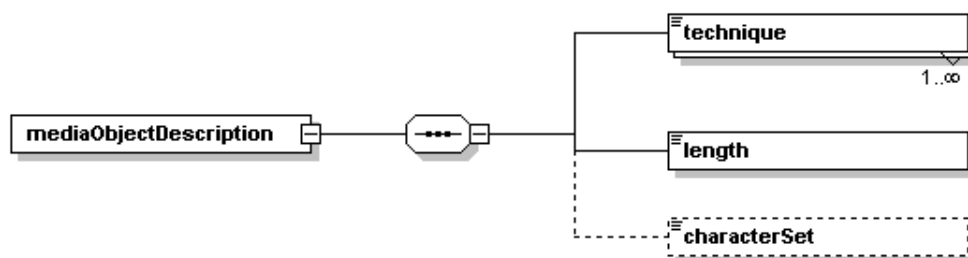
```
<xsd:element name="colourDepth" type="xsd:integer"/>
```

**Figure 52. XML Schema of the `colourDepth` element**

### 2.3.6 Description AMS

The Media Object Type Description AMS (MOT Description) is used to describe the Media Objects of type Description stored in the ARCO Database. In this section, the XML Schema implementing of the MOT Description AMS element set defined in [1], is described. The overall MOT Description AMS schema is presented in Figure 53. The schema consists of three elements. These elements are described below. For actual implementation see Appendix G.





**Figure 53. The overall XML Schema for Description AMS**

**Element: technique (MOT Description: Technique)**

The *technique* element implements the MOT Description AMS element *Technique* [1] (3.6.2.1). It contains information about the technique used to acquire Media Object. The *technique* element is a mandatory child element of the *mediaObjectDescription* element, and may occur multiple times. The type of the element is `xsd:string`. The value of the element can be only one from the predefined list: *OCR* or *Typing in an editing tool* (see Figure 54).

```
<xsd:element name="technique" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="OCR"/>
      <xsd:enumeration value="Typing in an editing tool"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 54. XML Schema of the technique element**

**Element: length (MOT Description: Length)**

The *length* element implements the MOT Description AMS element *Length* [1] (3.6.2.2). It provides the text length in chars of the description media object. The *length* element is a mandatory child element of the *mediaObjectDescription* element, and must occur only once. The type of the element is `xsd:long` (see Figure 55). The element is auto-generated by ACMA with the code name *MOT\_Description\_Length*.

```
<xsd:element name="length" type="xsd:long"/>
```

**Figure 55. XML Schema of the length element**

**Element: characterSet (MOT Description: Character set)**

The *characterSet* element implements the MOT Description AMS element *Character set* [1] (3.6.2.3). It contains the character set used for the text of the description media object. The *characterSet* element is an optional child element of the *mediaObjectDescription*

element, and it may occur only once. The type of the element is `xsd:string`. The value of the element must be one from the predefined list that can be found at the <http://www.iana.org/assignments/character-sets> web site.

The XML Schema for the `characterSet` element is provided in Appendix L.

### 2.3.7 3D Studio Max Project AMS

The Media Object Type 3D Studio Max Project AMS (MOT 3D Studio Max Project) is used to describe the Media Objects of type 3D Studio Max Project stored in the ARCO Database. In this section, the XML Schema implementing of the MOT 3D Studio Max Project AMS element set defined in [1], is described. The overall MOT 3D Studio Max Project AMS schema is presented in Figure 56. The schema consists of three elements. These elements are described below. For actual implementation see Appendix H.

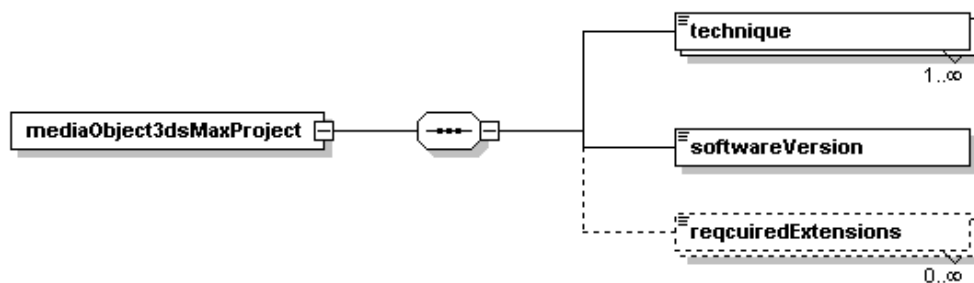


Figure 56. The overall XML Schema for 3D Studio Max AMS

#### Element: `technique` (MOT 3D Studio Max Project: Technique)

The `technique` element implements the MOT 3D Studio Max Project AMS element *Technique* [1] (3.6.3.1). It contains information about the technique used to acquire Media Object. The `technique` element is a mandatory child element of the `mediaObject3dsMaxProject` element, and may occur multiple times. The type of the element is `xsd:string`. The value of the element can be only one from the predefined list: *Image Based Modelling* or *Manual Modelling* (see Figure 57).

```
<xsd:element name="technique" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Image Based Modelling"/>
      <xsd:enumeration value="Manual Modelling"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Figure 57. XML Schema of the `technique` element

#### Element: `softwareVersion` (MOT 3D Studio Max Project: Software version)

The `softwareVersion` element implements the MOT 3D Studio Max Project AMS element *Software version* [1] (3.6.3.2). It contains the version of 3D Studio MAX software used to save

this file. The `softwareVersion` element is a mandatory child element of the `mediaObject3dsMaxProject` element, and it must occur only once. The type of the element is `allVersions` (see Figure 58).

```
<xsd:element name="softwareVersion" type="allVersions"/>
```

**Figure 58. XML Schema of the `softwareVersion` element**

The `allVersions` type is a simple type derived from the union of two simple types: `existingVersions`, `allPossibleVersions` (see Figure 59).

```
<xsd:simpleType name="allVersions">
  <xsd:union memberTypes="existingVersions allPossibleVersions"/>
</xsd:simpleType>
```

**Figure 59. XML Schema of the `allVersions` type**

`ExistingVersions`, is a simple type based on `xsd:string` that restricts the value of the `softwareVersion` to be one from the predefined directory: 3.0, 4.0, 4.2, 5.0 (see Figure 60).

```
<xsd:simpleType name="existingVersions">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="3.0"/>
    <xsd:enumeration value="4.0"/>
    <xsd:enumeration value="4.2"/>
    <xsd:enumeration value="5.0"/>
  </xsd:restriction>
</xsd:simpleType>
```

**Figure 60. XML Schema of the `existingVersions` type**

`AllPossibleVersions` is of type `xsd:string` to allow the element to accept any value.

**Element: `requiredExtensions` (MOT 3D Studio Max Project: Required extensions)**

The `requiredExtensions` element implements the MOT 3D Studio Max Project AMS element Required extensions [1] (3.6.3.3). It provides information about extensions and plug-ins needed to properly open and display this 3D Studio Max Project file. The `requiredExtensions` element is an optional child element of the `mediaObject3dsMaxProject` element, and it may occur only once. The type of the element is `xsd:string` (see Figure 61).

```
<xsd:element name="requiredExtensions" type="xsd:string" minOccurs="0">
```

**Figure 61. XML Schema of the `requiredExtensions` element**

### 2.3.8 VRML Model AMS

The Media Object Type VRML Model AMS (MOT VRML Model) is used to describe the Media Objects of type VRML Model stored in the ARCO Database. In this section, the XML Schema implementing of the MOT VRML Model AMS element set defined in [1], is described. The overall MOT VRML Model AMS schema is presented in Figure 62. The schema consists of five elements. These elements are described below. For actual implementation see Appendix I.

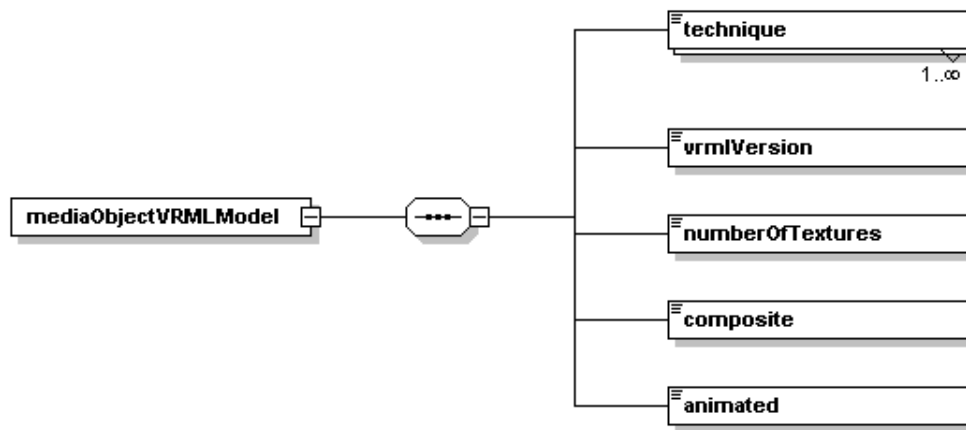


Figure 62. The overall XML Schema for VRML Model AMS

#### Element: technique (MOT VRML Model: Technique)

The technique element implements the MOT VRML Model AMS element *Technique* [1] (3.6.4.1). It contains information about the technique used to acquire Media Object. The technique element is a mandatory child element of the mediaObjectVRMLModel element, and may occur multiple times. The type of the element is xsd:string. The value of the element can be only one from the predefined list: *3DMAX export*, *Image based Modelling* or *Manual Modelling* (see Figure 63).

```

<xsd:element name="technique" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="3DMAX export"/>
      <xsd:enumeration value="Image based Modelling"/>
      <xsd:enumeration value="Manual Modelling"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
  
```

Figure 63. XML Schema of the technique element

#### Element: vrmlVersion (MOT VRML Model: VRML version)

The vrmlVersion element implements the MOT VRML Model AMS element *VRML version* [1] (3.6.4.2). It contains the version of VRML language used by the model. The vrmlVersion

element is a mandatory child element of the `mediaObjectVRMLModel` element, and it must occur only once. The type of the element is `xsd:string`. The value of the element can be only one from the predefined list: *1.0*, *2.0*, *97*, *200x* or *x3D* (see Figure 64). The `vrmlVersion` element is auto-generated by ACMA with the code name `MOT_VRML_Version`.

```
<xsd:element name="vrmlVersion">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="1.0"/>
      <xsd:enumeration value="2.0"/>
      <xsd:enumeration value="97"/>
      <xsd:enumeration value="200x"/>
      <xsd:enumeration value="x3D"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 64. XML Schema of the `vrmlVersion` element**

**Element: `numberOfTextures` (MOT VRML Model: Number of Textures)**

The `numberOfTextures` element implements the MOT VRML Model AMS element *Number of Textures* [1] (3.6.4.3). It provides the number of textures used by this VRML model. The `numberOfTextures` element is a mandatory child element of the `mediaObjectVRMLModel` element, and it must occur only once. The type of the element is `xsd:integer` (see Figure 65). The element is auto-generated by ACMA with the code name `MOT_VRML_NumberOfTextures`.

```
<xsd:element name="numberOfTextures" type="xsd:integer"/>
```

**Figure 65. XML Schema of the `numberOfTextures` element**

**Element: `composite` (MOT VRML Model: Composite)**

The `composite` element implements the MOT VRML Model AMS element *Composite* [1] (3.6.4.4). It indicates whether the model is composed of more than one VRML file. The `composite` element is a mandatory child element of the `mediaObjectVRMLModel` element, and it must occur only once. The type of the element is `xsd:boolean` (see Figure 66). The element is auto-generated by ACMA with the code name `MOT_VRML_Composite`.

```
<xsd:element name="composite" type="xsd:boolean"/>
```

**Figure 66. XML Schema of the `composite` element**

**Element: `animated` (MOT VRML Model: Animated)**

The `animated` element implements the MOT VRML Model AMS element *Animated* [1] (3.6.4.5). It indicates whether the model contains animated elements. The `animated` element is

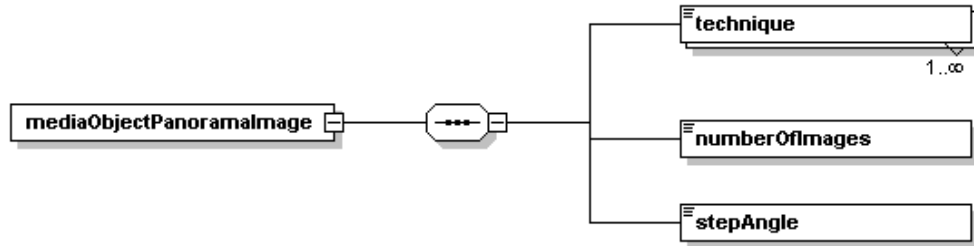
a mandatory child element of the `mediaObjectVRMLModel` element, and it must occur only once. The type of the element is `xsd:boolean` (see Figure 67). The element is auto-generated by ACMA with the code name `MOT_VRML_Animated`.

```
<xsd:element name="animated" type="xsd:boolean"/>
```

**Figure 67. XML Schema of the animated element**

### 2.3.9 Panorama Image AMS

The Media Object Type Panorama Image AMS (MOT Panorama Image) is used to describe the Media Objects of type Panorama Image stored in the ARCO Database. In this section, the XML Schema implementing of the MOT Panorama Image AMS element set defined in [1], is described. The overall MOT Panorama Image AMS schema is presented in Figure 68. The schema consists of three elements. These elements are described below. For actual implementation see Appendix J.



**Figure 68. The overall XML Schema for Panorama Image AMS**

#### **Element: technique (MOT Panorama Image: Technique)**

The `technique` element implements the MOT Panorama Image AMS element *Technique* [1] (3.6.5.1). It contains information about the technique used to acquire Media Object. The `technique` element is a mandatory child element of the `mediaObjectPanoramaImage` element, and may occur multiple times. The type of the element is `xsd:string`. The value of the element can be only one from the predefined list: *Automated Image Processing* or *Human Image Processing* (see Figure 69).

```
<xsd:element name="technique" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Automated Image Processing"/>
      <xsd:enumeration value="Human Image Processing"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 69. XML Schema of the technique element**

**Element: numberOfImages (MOT Panorama Image: Number of images)**

The numberOfImages element implements the MOT Panorama Image AMS element *Number of images* [1] (3.6.5.2). It provides the number of images included in this panorama view. The numberOfImages element is a mandatory child element of the mediaObjectPanoramaImage element, and it must occur only once. The type of the element is xsd:integer (see Figure 70). The element is auto-generated by ACMA with the code name MOT\_PanoramaImage\_NumberOfImages.

```
<xsd:element name="numberOfImages" type="xsd:integer"/>
```

**Figure 70. XML Schema of the numberOfImages element**

**Element: stepAngle (MOT Panorama Image: Step angle)**

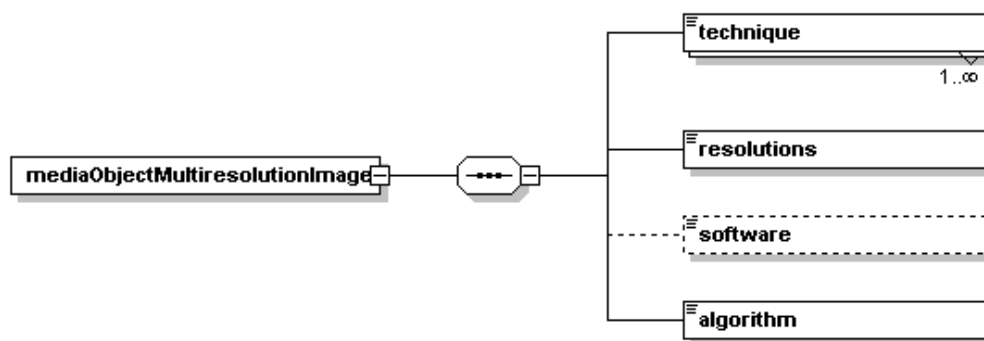
The stepAngle element implements the MOT Panorama Image AMS element *Step angle* [1] (3.6.5.3). It contains the step angle between images. The stepAngle element is a mandatory child element of the mediaObjectPanoramaImage element, and it must occur only once. The type of the element is xsd:float (see Figure 71). The element is auto-generated by ACMA with the code name MOT\_PanoramaImage\_StepAngle.

```
<xsd:element name="stepAngle" type="xsd:float"/>
```

**Figure 71. XML Schema of the stepAngle element**

### 2.3.10 Multiresolution Image AMS

The Media Object Type Multiresolution Image AMS (MOT Multiresolution Image) is used to describe the Media Objects of type Multiresolution Image stored in the ARCO Database. In this section, the XML Schema implementing of the MOT Multiresolution Image AMS element set defined in [1], is described. The overall MOT Multiresolution Image AMS schema is presented in Figure 72. The schema consists of four elements. These elements are described below. For actual implementation see Appendix K.



**Figure 72. The overall XML Schema for Multiresolution Image AMS**

**Element: technique (MOT Multiresolution Image: Technique)**

The technique element implements the MOT Multiresolution Image AMS element *Technique* [1] (3.6.6.1). It contains information about the technique used to acquire Media Object. The technique element is a mandatory child element of the mediaObjectMultiresolutionImage element, and may occur multiple times. The type of

the element is `xsd:string`. The value of the element can be only one from the predefined list: *ACMA wizard*, *Automated image processing* or *Human image processing* (see Figure 73).

```
<xsd:element name="technique" maxOccurs="unbounded">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="ACMA wizard"/>
      <xsd:enumeration value="Automated image processing"/>
      <xsd:enumeration value="Human image processing"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

**Figure 73. XML Schema of the technique element**

**Element: resolutions (MOT Multiresolution Image: Resolutions)**

The resolutions element implements the MOT Multiresolution Image AMS element *Resolutions* [1] (3.6.6.2). It provides a list of available image sizes in pixels. The resolutions element is a mandatory child element of the `mediaObjectMultiresolutionImage` element, and it must occur only once. The type of the element is a list of `xsd:string` elements with a regular expression enforcing values in format *widthxheight* (see Figure 74). The resolutions element is auto-generated by ACMA with the code name `MOT_MultiresolutionImage_Resolutions`.

```
<xsd:element name="resolutions">
  <xsd:simpleType>
    <xsd:list>
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="\d+x\d+"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:list>
  </xsd:simpleType>
</xsd:element>
```

**Figure 74. XML Schema of the resolutions element**

**Element: software (MOT Multiresolution Image: Software)**

The software element implements the MOT Multiresolution Image AMS element *Software* [1] (3.6.6.3). It provides the name of application used for generating the Multiresolution image. The software element is an optional child element of the



`mediaObjectMultiresolutionImage` element, and it may occur only once. The type of the element is `xsd:string` (see Figure 75).

```
<xsd:element name="software" type="xsd:string" minOccurs="0"/>
```

**Figure 75. XML Schema of the software element**

**Element: algorithm (MOT Multiresolution Image: Algorithm)**

The `algorithm` element implements the MOT Multiresolution Image AMS element *Algorithm* [1] (3.6.6.4). It provides information about the algorithm used for rescaling the original image. The `algorithm` element is an optional child element of the `mediaObjectMultiresolutionImage` element, and it may occur only once. The type of the element is `xsd:string` (see Figure 76).

```
<xsd:element name="algorithm" type="xsd:string" minOccurs="0"/>
```

**Figure 76. XML Schema of the algorithm element**

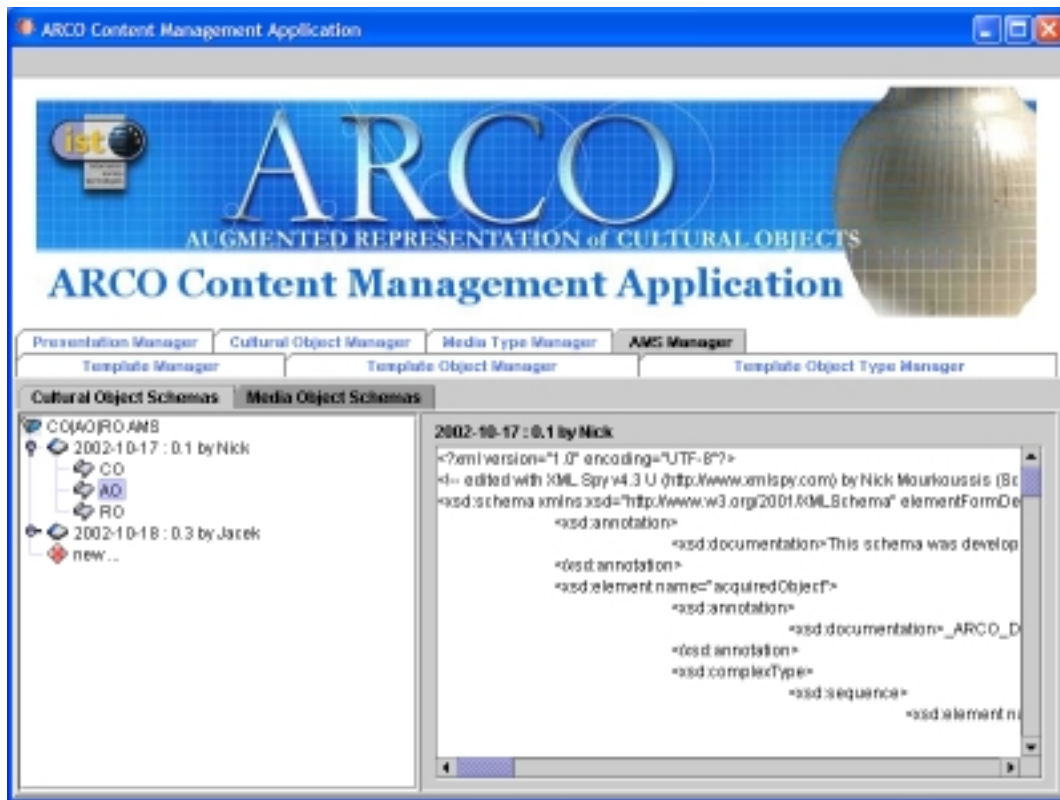
## 2.4 AMS Schema Manager

One of the powerful features of the ARCO system is the ability to use new versions of AMS schema as the AMS specification evolves. New AMS specifications can be developed at museum pilot sites and loaded into the ARCO database without the need to modify ARCO database structure or the ACMA or ARIF tools.

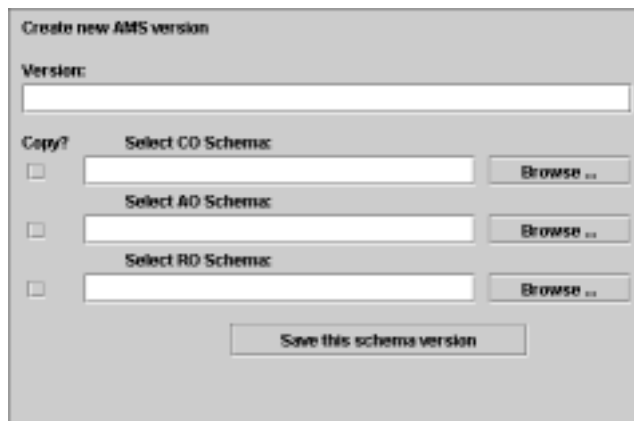
Since the AMS specification may change over time, the ARCO database must keep historical versions of the AMS specification in order to support cultural objects that use older versions of the AMS specification. Otherwise, expensive conversion of XML metadata descriptions would be required. Due to the fact that the XML Schemas of the AMS metadata are stored in the database, every Cultural Object and Media Object may be associated with a correct version of the XML Schema describing structure of its metadata.

In order to be able to manage XML Schemas in the ARCO database in a user-friendly way, a special tool – the *AMS Schema Manager* has been designed and implemented. The ARCO AMS Schema Manager is integrated within the ACMA – ARCO Content Management Application. A screenshot of the AMS Schema Manager is presented in Figure 77.

There are two basic types of XML Schemas stored in the ARCO Database and managed by the ARCO AMS Manager: *Cultural Object Schemas* and *Media Object Schemas*. They are represented by two tab panels in the AMS Schema Manager Window. Each tab panel contains two elements: schema version tree on the left side and preview/add panel on the right side.



If a user tries to delete a version that is used by one or more Cultural Objects, an appropriate error message is displayed and the operation is cancelled.



**Figure 78. Adding new AMS schema in the ARCO AMS Schema Manager**

## 2.4.2 Media Object AMS Manager

A powerful feature of the ARCO system is the extensibility of the set of supported Media Object Types. New Media Object Types may be added to the system without modifying the database structure or ACMA and ARIF tools.

Since a specific AMS schema may be associated with each Media Object Type (MOT AMS), the Media Object AMS Manager must provide support for both:

- adding new types of MOT AMS schemas, and
- adding new versions for existing MOT AMS schemas.

A screenshot of the Media Object panel of the AMS Manager is presented in Figure 79. The tree on the left side of the window displays all MOT AMS schemas and their versions. The tree is organized as follows. Root element ('*MOType AMS*') contains the general MO AMS Schema (used for all Media Object Types) and all MOT AMS schemas (usually, one for each Media Object Type), described by the name of their XSD root node, and a special '*new schema...*' element. Each AMS schema node contains nodes that represent versions of this MOT AMS schema (or the GeneralMediaObjectAMS Schema). The nodes display the creation dates and version names of the schema versions. Each schema node contains also a special '*new version...*' node.

By selecting a leaf node corresponding to an AMS schema version a user may display the AMS schema version in the right panel. By selecting one of the special '*new...*' node, a user may add a new schema or a new schema version.



**Figure 79. AMS Schema Manager – Media Object AMS**

A user may perform three types of operations:

- create a new AMS schema – by the use of the ‘*new schema...*’ node,
- create a new AMS schema version – by the use of the ‘*new version...*’ node, or
- delete a schema or a schema version – by the use of the “delete” option accessible in the right-click popup menu.

**Figure 80. Creating a new AMS schema in MO AMS Manager**

When the “new schema” operation is selected, a special form is displayed. The form allows the use to enter a new schema name and description (Figure 80).



**Figure 81. Creating a new version of AMS schema in MO AMS Manager**

When the “new schema version” operation is selected, another form is displayed that allows to enter the version name and select the XML Schema file (\*.xsd) containing new schema definition (Figure 81).

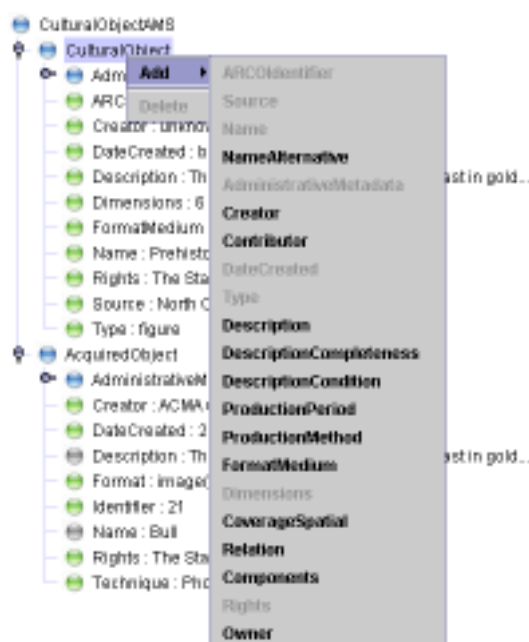
## 2.5 AMS Metadata Editor

In order to make the process of editing the metadata accompanying objects in the ARCO database as efficient as possible a special tool – *AMS Metadata Editor* has been developed. The AMS Metadata Editor is integrated with ACMA Cultural Object Manager as an embedded application and is used for editing metadata of both Cultural Objects (Acquired Objects and Refined Objects) and Media Objects (all types), see Figure 82. The AMS Metadata Editor allows a user to manage AMS metadata documents. The XML documents are presented as trees, where the user can add or delete nodes, and edit their values.



Figure 82. ARCO AMS Metadata Editor

To add a new node, a user has to right click on the parent node and choose 'Add' menu item from the popup menu. The list of allowed nodes is displayed (Figure 83). The list is generated according to the AMS schema definition. Some of the nodes on the list may be disabled because maximum occurrence limit defined in the AMS schema is reached, or element group has a 'choice' model and one of the possible elements has been already created. Elements that are not optional are created automatically when a new AMS document is built.



**Figure 83. Adding a new element in AMS Metadata Editor**

For deleting an element also the popup menu is used – item ‘Delete’. However, not every element can be deleted – because the minimum occurrences limit defined in AMS schema is taken into consideration.

Values for metadata elements are entered using a text field below the metadata tree. When entered, a value is validated using AMS definition, and if it is not correct, a dialog window with appropriate error message appears.



**Figure 84. Additional functions of the AMS Metadata Editor**

There are also two special functions available in the popup menu on the root node: ‘Show XML’ for displaying textual representation of AMS XML and ‘Save AMS’ for saving AMS metadata XML to a file (Figure 84).

The metadata editor uses icons with different colors for different types of elements:

	Blue	Complex elements that may contain other elements, but no values
	Green	Simple leaf element – may contain a value
	Gray	Auto-generated element – value is generated by the application (read-only)
	Yellow	Dictionary element – values are taken from a dictionary stored in ARCO Database
	Red	Element that does not conform to the assigned AMS schema, this should be treated as a warning that something is wrong - AMS XML document must always conform to the assigned AMS schema.



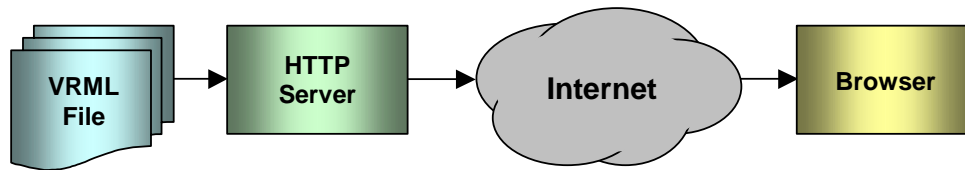


### 3. X-VRML Technology

#### 3.1 Introduction to the X-VRML Language

The VRML standard provides methods of describing contents of virtual scenes. In its current form, it enables building passive virtual reality systems, i.e. systems where the virtual reality is employed to visualize some pre-designed virtual environments in a three-dimensional way. This enables using VRML for presenting pre-designed models and simple animations, but prevents it from being used in more advanced applications.

In standard VRML systems, descriptions of virtual scenes are stored in ASCII form in flat files. Since the descriptions are static, once created a virtual scene may be displayed to the user only in exactly the same, previously prepared form. The architecture of a conventional VRML system is presented in Figure 85.



**Figure 85. Architecture of a conventional VRML system**

A set of VRML files is stored on an HTTP server. Each VRML file describes a set of virtual objects. Virtual scenes are composed of one or more VRML files. Browsers can access the virtual scenes by providing appropriate URLs of top-level VRML files. As a response, the HTTP server sends the requested files to the browser. The browser displays the virtual scene and the user may play with it. A user can move from one virtual scene to another by following a link or entering a new URL address.

In such systems, the description of the virtual scenes is final – the set of virtual objects contained in the scene, their positions, and values of attributes stored in the files are constant. Such systems have several serious disadvantages that are shortly summarized below.

There is no possibility to select parts of the virtual scene (information) that should be visualized. There is no possibility to use any data selection criteria, i.e., a user cannot specify what parts (geometrical or logical) of the virtual scene should be displayed. As a result, the entire virtual scene description must be sent to the client's browser at once. There is no possibility to change appearance of a virtual world or a part of it by changing some of its creation parameters – creation of the description is performed once. Since the user cannot alter this process, no influence on the final form of the virtual world is possible. There is no possibility to automatically update the virtual world data. Data used for creation of the virtual scene are up-to-date when the virtual scene is created. At the time a user accesses the virtual scene, the data may no longer be valid. Such a virtual reality system cannot be used for visualization of changing data. Since the process of editing a virtual world is based only on its geometry, it may be inefficient and time consuming. The security model is simplistic when data are stored in flat files – the smallest unit of access control is a file.

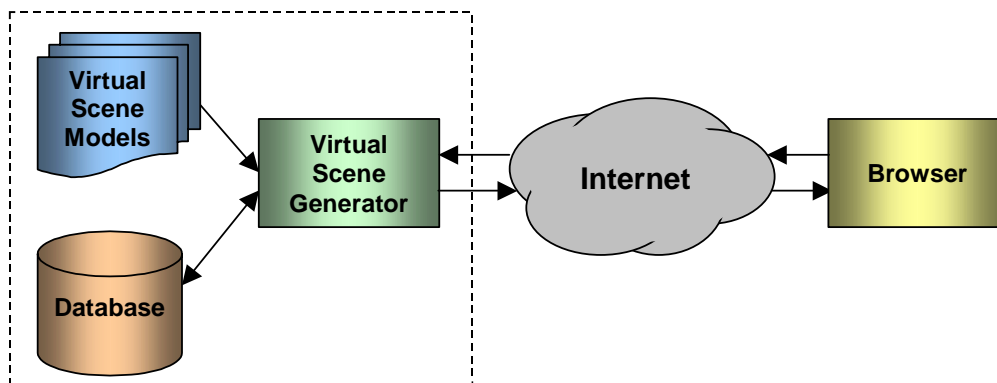
Another serious limitation of the VRML language is the lack of convenient database access methods. The problem of accessing databases from VRML programs has been investigated by the VRML Database Working Group [8][9]. A solution proposed in [10][11][12] has been finally accepted as Recommended Practices for accessing databases from VRML. It consists of two main elements: SQL Scripting that provides methods of accessing databases from virtual

scenes at the run-time, and ServerRedirect that enables restoring the virtual scene state when it is accessed by a user. Despite the fact that the ServerRedirect solution uses special processing software on the server side, it is very limited and allows only controlling values of some types of fields in VRML nodes. Access to fields that are not directly supported is difficult. A serious limitation of the ServerRedirect solution follows from the fact that only values of fields of existing nodes can be retrieved from the database. This means that both virtual scene structure and visualization method must be hard-coded in the VRML program and only attributes of existing elements can be influenced by data coming from a database. To enable building active database applications of virtual reality, a much more convenient and efficient method of accessing databases is required.

To overcome the above-mentioned problems the ARCO system will use, adapt and further extend a novel virtual reality modelling language called X-VRML. The X-VRML language has been designed by one the project Partners [13][15][16][17] to enable dynamic modelling of virtual reality [14]. The dynamic modelling technique enables developing active database-driven virtual reality applications by building parameterised models of virtual scenes that constitute an application, and dynamic generation of the instances of virtual scenes based on the models and current values of model parameters, query provided by a user, user privileges, user preferences, and the current system state.

The X-VRML language provides convenient access to databases. The data retrieved from the database can affect all aspects of the dynamically generated virtual scenes – the contents, the visualization methods, and the structure. Furthermore, X-VRML offers powerful parameterisation of models and programming concepts known from procedural languages like loops, conditions and variables, which combined with the declarative VRML approach result in a powerful programming tool. Moreover, X-VRML supports object-oriented programming style by enabling inheritance hierarchies of classes that can be used in the virtual scene models. The X-VRML language is based on XML, which is currently the de-facto standard for creating new languages in this domain. This also combines well with other XML solutions used in the ARCO system.

Architecture of a model-based virtual reality system is presented in Figure 86. Every time a user wants to access a virtual scene or a selected part of it, a request is sent to the server. The server reads the virtual scene model, interprets it, and creates “on-the-fly” an appropriate description of the virtual scene. The user can use such a virtual scene in exactly the same way as a “standard” virtual scene stored in a file. During the process of virtual scene generation, however, multiple factors may be taken into consideration to influence its final form. These factors include selection criteria, user preferences – provided by the user or taken from a data repository, user privileges, creation method, and up-to-date data read from one or several databases.



**Figure 86. Architecture of a model-based virtual reality system**

The virtual scene generator is a generic tool independent of the particular virtual scene or application. The same virtual scene generator can be used to generate different virtual scenes when provided with different models. One virtual scene generator may be used to serve multiple VR scenes. Such an approach enables use of complex, highly optimised implementation of the virtual scene generator without extending the amount of work required to develop a single model.

3.2 Language Overview

The X-VRML language is based on XML [18]. An X-VRML model is an XML representation of an algorithm that generates virtual scenes. The model is a program built of X-VRML commands. The program can read data from a database and use values of model parameters as input. Examples of commands are: `set` to assign a value to a variable, `for` implementing a numerical loop, `if/then/else` implementing conditional statements [13], and `class/instance` representing classes and their instances [15]. The X-VRML commands are encoded in XML and placed inside the text of the scene description written in VRML, X3D, or HTML. The fact that the target language (e.g., X3D) contains XML tags is not an obstacle due to special processing of the source X-VRML file: all XML tags that are not known to the X-VRML processing unit (e.g., use a different namespace) are simply ignored and included in the outcome as fragments of the scene description.

In X-VRML, empty XML elements represent single-line commands such as `<set/>` or `<insert/>`. Non-empty elements represent block-statements like loops, conditions, database queries, iterations, etc. Examples of non-empty elements are `<for> ... </for>` and `<db_query> ... </db_query>`. The non-empty elements can contain fragments of scene description and X-VRML code. The code located inside a repeating element is interpreted multiple times. Parameters for the command execution are provided in the form of values of element attributes. In X-VRML, all parameters are expressions and are evaluated prior to command interpretation.

Below, an example of a numerical loop is presented. The loop uses an “i” control variable, varying from 0 to the value of 2k+10 (“k” is an X-VRML variable) and incremented by 1:

```
<for name="i" from="0" to="2*$k+10" step="1">
...
</for>
```

A comparison of Java and X-VRML representations of the same simple algorithm is shown in Table 1.

<pre>for (int i = 0; i &lt;= 10; i++) {     int x = i*i;     if (i &lt; 5)     {         ...     }     else     {         ...     } }</pre>	<pre>&lt;for name="i" from="0" to="10" step="1"&gt;   &lt;set name="x" value="\$i*\$i"/&gt;   &lt;if condition="\$i&lt;5"&gt;     &lt;then&gt;       ...     &lt;/then&gt;   &lt;else&gt;     ...   &lt;/else&gt; &lt;/if&gt; &lt;/for&gt;</pre>
Java Version	X-VRML Version

**Table 1. Comparison of Java and X-VRML versions of the same algorithm**

### 3.3 Modularisation of X-VRML

Functionality of the X-VRML language is divided into modules. Three different modules of X-VRML are commonly used in server-side system: Core Module, Object-Oriented Module, and Database Module.

The X-VRML Core Module provides basic language functionality that can be used in most application domains. The basic functionality allows assigning values to variables, inserting calculated values of expressions to the output scene description, conditional interpretation of fragments of the X-VRML code, loops, and nesting X-VRML models [13].

The X-VRML Object-Oriented Module provides methods of implementing classes, building class inheritance hierarchies, and using instances of classes [15].

The X-VRML Dynamic Module is a set of X-VRML commands allowing implementation of dynamic virtual scene models, i.e. models of scenes whose structure changes during the virtual scene run-time. The change in the scene structure can be a result of the dynamism coded in the scene model, the user interaction, or new data read from a database [14].

The X-VRML Database Module contains language constructs that allow access to miscellaneous data sources including relational and object-oriented database systems. Access includes both retrieval and updating of the database contents [15].

One of the fundamental concepts of X-VRML is extensibility. New elements – performing more advanced tasks – can be easily added to the language. Adding new elements can be beneficial in applications that use complex processing or are time-critical. Usually interpretation of a single complex element is faster than interpretation a long X-VRML program composed of simple elements. Extensibility allows to create additional, specialized X-VRML modules with functionality specific for particular domain or application.

An additional module of X-VRML has been developed in the ARCO project. The module contains elements simplifying retrieval of ARCO specific data from the ARCO database (ARIF Folders, Cultural Objects, Media Objects, AMS).

### 3.4 X-VRML Core Module

#### 3.4.1 Overview

The X-VRML Core Module provides basic language functionality that can be used in most application domains. The basic functionality allows assigning values to variables, inserting calculated values of expressions to the output VRML file, conditional interpretation of fragments of the X-VRML code, loops, and nesting X-VRML files. In the remainder of this section, the basic elements of the X-VRML Core Module are described.

#### 3.4.2 Variables and expressions

The X-VRML language allows to use variables. Variables can be set inside X-VRML programs by the use of X-VRML commands (e.g., `set`, `for`, `iteration`, or `db_query`), can be read from a configuration file, or can be provided in the URL. Values of variables provided in the URL can be statically specified by a hypertext reference or dynamically set on the client side (e.g., in an HTML form filled by a user).

In X-VRML, all values of element attributes are treated as expressions and are evaluated prior to element interpretation. Expressions can contain:

- constant numerical, textual, and Boolean values,
- variable references,
- operators, and
- functions.

### 3.4.3 Assigning values to variables

X-VRML allows assigning values to variables. Assignment of a value to a variable may be accomplished by the use of the `set` command. The syntax of the `set` element is as follows:

```
<set name="..." value="..." />
```

The `set` element is an empty XML element (not containing X-VRML code). The `set` element has two mandatory attributes:

- `name` represents the name of the variable. The variable can be later referenced by the use of this name.
- `value` represents the value that should be assigned to the variable. This can be as well a simple constant value as an expression that must be evaluated prior to the assignment.

### 3.4.4 Inserting values

Calculated values of X-VRML expressions can be included into the outcome VRML file by the use of the `insert` element.

The syntax of the `INSERT` element is as follows:

```
<insert value="..." type="..." />
```

The `insert` tag is replaced by the interpreter with the value calculated from the contents of the `value` attribute. This attribute can contain a simple constant value, a variable reference, or an expression that must be evaluated by the interpreter prior to inserting.

The optional `type` attribute is used to cast the result of value calculation to a specific type. Examples of possible final types include `INT` (for integer values), `FLOAT` (for floating point numerical values), and `BOOL` (for Boolean values).

### 3.4.5 Conditional Statements

X-VRML allows conditional parsing of fragments of X-VRML code. Conditional statements are expressed by `if` elements. The non-empty `if` element contains the part of the X-VRML code that is included in the outcome only if the specified condition is satisfied, and the optional part that is included when the specified condition is not satisfied.

The syntax of the `if` element is as follows:

```
<if condition="...">  
  <then>  
    ...  
  </then>  
  <else>  
    ...  
  </else>  
</if>
```

The `if` element can contain only `then` and `else` elements. The `if` element has only one attribute: `condition`. This mandatory attribute specifies an expression string that is evaluated to a Boolean value. If the value of the `condition` attribute evaluates to `TRUE`, the `then` element is parsed. Otherwise, the `else` element is parsed.

### 3.4.6 Loops

Loops are one of the basic components of X-VRML. They are particularly helpful for coding elements with repeating structure or created in result of some repeating mathematical calculations. There are three types of loops in X-VRML Core Module:

- `for` – loop repeated with a numerical control variable varying from a beginning to an ending value, every time changed by a step value,
- `iteration` – repeated for each value in a specified list of values, and
- `while` – repeated as long as a condition is satisfied.

#### For Loop

The `for` loop may be used to repeat a fragment of code an arbitrary number of times with a numerical control variable. The variable takes values from a specified range of values (defined by `from` and `to` attributes). During the first loop traversal it is assigned the value defined by the `from` attribute. During each of the subsequent loop traversals its value is being modified by the value defined in the `step` attribute. When the value declared in the `to` attribute is exceeded, the loop is not being repeated anymore.

The syntax of the `for` element is as follows:

```
<for name="..." from="..." to="..." step="...">  
...  
</for>
```

The `for` element contains the X-VRML code that should be traversed a number of times with the changing value of the control variable.

The `for` element has four attributes: `name`, `from`, `to`, and `step`. The meaning of the attributes is the following:

- `name` is mandatory and represents the name of the loop control variable. The value of the loop control variable may be accessed inside the loop as any other variable.
- `from` is optional and represents the initial value for the loop control variable. During the first loop traversal, the loop control variable is set to the value defined by this attribute. After each loop traversal the value of the control variable is being modified by the value specified by the `step` attribute. Default value of the `from` attribute is "1".
- `to` is mandatory and represents the ending value for the control variable. When the control variable exceeds the value specified by the `to` attribute, the loop is not repeated anymore.
- `step` is optional and represents the value by which the control variable is modified after each loop traversal. The value of the `step` attribute may be positive indicating that the control variable should be increased after each loop traversal or negative indicating that the control variable value should be decreased. If the `step` attribute is omitted, the default value "1" is taken.

### Iteration

The iteration loop in X-VRML is a loop that is repeated an arbitrary number of times, once for each value specified in a list of values. For each of the loop traversals, the control variable is set to the next value in the list.

This control structure is particularly useful for repeating a fragment of code in case when the list of occurrences is known during the design time. Nevertheless, since the list can contain not only constant values, but also expressions that are evaluated before the values are assigned to the control variable, the iteration loop can be used also for values that are calculated during the processing time.

The syntax of the iteration element is as follows:

```
<iteration name="..." list="...,...,...">
...
</iteration>
```

The iteration tag denotes a non-empty XML element containing the X-VRML code that should be repeated a number of times, once for each value in the value list.

The iteration element has two attributes: `name` and `list`.

- `name` is mandatory and represents the name of the loop control variable. The value of the control variable may be accessed inside the loop.
- `list` is also mandatory and represents the list of values for the loop control variable. During the first traversal of the loop, the control variable is set to the first element from the list. During each of the subsequent traversals, the control variable is set to the next element on the list. The list can contain as well numeric as non-numeric values. These can be constant values entered during the design-time or expressions. Values of expressions are calculated before assigning them to the loop control variable. Values in the list can be provided as a coma-separated list or an expression evaluating to a list.

### While Loop

The while element is a loop that is repeated as long as a condition is satisfied. The loop does not have any loop control variable. The programmer must organize the control inside the loop appropriately to ensure that the loop eventually exits. The loop condition is provided in the `condition` attribute.

The syntax of the while element is as follows:

```
<while condition="...">
...
</while>
```

The while tag denotes a non-empty XML element containing the X-VRML code that should be repeated as long the expression specified in `condition` attribute evaluates to value "TRUE".

The while loop has one mandatory attribute: `condition` – it is an expression evaluating to a Boolean value.

### 3.4.7 Nesting X-VRML Files

X-VRML allows including X-VRML files in other X-VRML files. There are two elements in X-VRML that can be used to accomplish this task - `inline` and `import`.

The `inline` element is a low-level element interpreted before the parsing process of the XML document tree starts. This element can be used to include any fragments of the X-VRML scene model (VRML, X-VRML, or arbitrary sub-fragments). Since the element is interpreted prior to the document parsing, the document reference must be constant (it cannot use variables since the variables have undefined values). All variables set in the main X-VRML file are visible in the included files, and all variables set in the included files are visible in the including file.

As opposed to `inline`, the `import` element is interpreted during the main X-VRML document parsing. As a result, this element can use parameterised file references. In addition, the variables visibility can be controlled by the use of special `in` and `out` attributes. The files that are included by the use of the `import` element must be valid X-VRML models.

In both cases, the included X-VRML file may include other X-VRML files – by the use of either of the two methods. The depth of the inclusion graph is not limited. The inclusion graph may contain cycles as long as the models are parameterised appropriately to ensure that the process of inclusion is finite.

The syntax of the `inline` element is as follows:

```
<inline file="..." />
```

The `inline` element has one mandatory attribute: `file`, which indicates location and name of the X-VRML file to be included.

The syntax of the `import` element is as follows:

```
<import url="..." in="..." out="..." />
```

The `import` element has one mandatory attribute: `url`, which indicates location of the X-VRML file to be included, and two optional attributes: `in` and `out`.

The `in` and `out` parameters are used to control the variables visibility. If the `in` parameter is set to “TRUE”, variables set in the including file are visible in the included file. If the `out` parameter is set to TRUE, variables set in the included file are visible in the including file. Otherwise, the variables are not visible.

## 3.5 The X-VRML Database Module

### 3.5.1 Overview

The X-VRML Database Module contains language elements that allow access to miscellaneous data sources including relational and object-oriented database systems. The access means either retrieving or updating data in the database. Depending on system architecture and configuration, the database access may be accomplished either once during the virtual scene generation process or continuously at the run-time.

Since connections to multiple databases are allowed even from one X-VRML model, the connections must be explicitly established. Before the interpreter connects to a database, it must be provided with connection parameters. These parameters include network address of the computer that the database is running on, port number, database name, and user name/password. In order to allow connections to different types of data sources, the interpreter must also be provided with the database driver name.



### 3.5.2 Connecting to Databases

In X-VRML, a connection to a database is established by the `db_connect` element. The required connection details are provided in two parameters: `connection_string` and `driver`. If the parameters are omitted, the necessary connection information is taken from two X-VRML variables: `connection_string` and `connection_driver`. The syntax of the `db_connect` element is as follows:

```
<db_connect connection_string="..." driver="...">
...
</db_connect>
```

In some implementations, the X-VRML interpreter may be permanently connected to a default data source. In such case, use of the `db_connect` element is not required, unless the system must connect to a data source different from the default one. If the `db_connect` element is required, it must include all database access elements. More than one `db_connect` element may appear in one X-VRML model allowing retrieval and update of data from/in more than one database.

If there are database access commands in X-VRML extension modules, they use the same connection mechanism provided by the `db_connect` element.

### 3.5.3 Retrieving Data from Databases

Retrieving data from databases is one of the most important features of the X-VRML language. This task may be accomplished by the use of a special `db_query` element. To simplify the process of retrieving data and constructing the output scene description, the `db_query` element has been designed to behave as a special kind of loop. The loop is being repeated for each row or object (depending on the type of the database used) obtained from a database as a result of query execution. The values retrieved from a database are assigned to a set of loop control variables.

In the element start-tag, a list of names of variables and the database SQL query are specified. The number of names of variables should be equal to the number of attributes retrieved from the database by the query. For each loop traversal, the retrieved values are assigned to variables identified by the provided list of names. The syntax of the `db_query` element is the following:

```
<db_query names="...,...,..." sql="...">
...
</db_query>
```

The `db_query` element has two mandatory attributes: `names` and `sql`. The `names` attribute contains a coma-separated list of names of the loop control variables. The `sql` attribute represents the text of the SQL query to be executed in the database. Since the `sql` attribute is an X-VRML expression, it may use X-VRML variables.

In a system configuration, where the X-VRML interpreter is not permanently connected to a database, the query loop must be located inside a `db_connect` element. A single connection established by one `db_connect` element may be used by more than one `db_query` element.

The `db_query` element is the most basic method of accessing data in databases from an X-VRML model. In advanced database applications of virtual reality other, more sophisticated methods of retrieving data from databases may be used (cf. Section 3.7).

### 3.5.4 Updating Databases

An important feature of the X-VRML language is a possibility of updating databases. Updating databases may be performed either during the virtual scene generation process – in such case it can be used mostly for statistical purposes, or continuously in run-time of the virtual scene – in such case it can be used for providing persistency to virtual worlds. Updating databases is accomplished in X-VRML by the use of a special `db_update` element. The syntax of this element is the following:

```
<db_update sql="..." />
```

The `db_update` element has one mandatory attribute: `sql` that contains the SQL command to be executed. The `sql` attribute is an X-VRML expression that is evaluated before the element is interpreted allowing use of X-VRML variables in SQL commands.

In a system configuration, where the X-VRML interpreter is not permanently connected to a database, the `db_update` element must be located inside a `db_connect` element.

## 3.6 X-VRML Object-Oriented Module

### 3.6.1 Concept

One of the major drawbacks of the current VRML design is the lack of object-oriented features like classes, inheritance hierarchies, and instances of classes. Use of object-oriented features simplifies the process of designing virtual worlds; it enables creation of libraries of frequently used classes and shortens the code required for encoding virtual scenes.

Classes in conventional object-oriented systems encapsulate object attributes and methods. In virtual reality systems, a new element that classes must deal with is geometry. Due to specific characteristics of the class geometry, it has been distinguished as a new major component constituting the X-VRML class.

The presence of geometry in the class definition influences the concept of class inheritance. Geometry can be modified by a subclass in the same way as attributes or methods in conventional object-oriented systems. In particular, the following cases are possible:

- a subclass can modify the geometry of its superclass, while inheriting attributes;
- an abstract superclass can use abstract components that can be defined in subclasses. A subclass may be non-abstract only if it defines all abstract components inherited from all its superclasses and does not use abstract components by itself;
- a subclass can extend the geometry of a superclass by inserting super-class implementation somewhere in the context of the class implementation; and
- a subclass can combine the geometry of a set of superclasses by providing space transformations for each implementation of the superclass geometry.

To meet the above-listed requirements, an explicit inheritance of class geometry has been used. It means that superclass implementation must be explicitly referenced in subclass implementation if it is going to be used by the subclass.

### 3.6.2 Defining X-VRML classes

An X-VRML class is defined by a set of XML elements containing X-VRML code. The syntax of a class definition is the following:

```
<class name="..." extends="...",..." abstract="...">
```

```
<interface>
  <attr_decl name="..." def_value="..." />
  ...
</interface>
<implementation>
  ...
</implementation>
</class>
```

The `class` element has three attributes:

- `name` is mandatory and provides the name of the class.
- `extends` is optional and represents names of class or classes this newly created class extends. The value "NONE" indicates that the class does not inherit from other classes. "NONE" is the default value.
- `abstract` is optional and indicates whether the class is abstract or non-abstract. If a class is abstract, it cannot be instantiated. Default value for the `abstract` attribute is "FALSE", which means, that if the attribute is omitted, the class is assumed to be non-abstract.

The `class` element may contain only two other elements, `interface` and `implementation`.

The `interface` element is optional and contains the interface part of the class definition. It contains `attr_decl` empty elements that are declarations of the attributes. The syntax of the `attr_decl` element is the following:

```
<attr_decl name="..." def_value="..." />
```

The `attr_decl` element has two mandatory attributes:

- `name` denotes the name of the attribute being declared. The attribute may be referenced inside the class implementation by the use of this name,
- `def_value` denotes the default value of the attribute. If an object does not specify the value of the attribute, the default value will be used.

If a class inherits from another class (`extends` it), it must declare only the attributes that are new or overridden in this new class. The attributes that are inherited from a superclass do not have to be listed in a subclass.

The `implementation` element contains the actual X-VRML implementation of the class. The implementation part of the class definition, after processing by the X-VRML processor, should be conformant to the VRML syntax (or any other target syntax).

If a class does not inherit from other X-VRML classes, the implementation part should contain the whole code of the class. If the class extends other classes, the implementation may contain the whole code of class or only parts of the code – parts that are defined, or added in this new class.

A class defined in X-VRML can be abstract. Abstract class is a class that is not fully defined and thus cannot be instantiated. An abstract class can use components that are not defined. These components can be specified later in subclasses. A subclass that defines all of its superclass components may be non-abstract. If it specifies only a subset of these components, it remains abstract.

Components in abstract classes are denoted by the `component` element. It is an empty tag. The syntax of the `component` element is the following:

```
<component name="..." />
```

The `component` tag requires one attribute: `name` that represents the component identifier. It can be used by subclasses to define the component implementation.

A subclass of an abstract class may define a component used in its superclass by the use of a `define` element. It has the following syntax:

```
<define name="..." class_name="...">
...
</define>
```

The `define` element is a non-empty XML element containing the X-VRML code that defines implementation of a component used by an abstract superclass. It has the mandatory attribute `name`, which provides the identifier of the component that is being defined. The code contained in a `define` element is used in place of a `component` element of the same name in a superclass implementation.

An optional `class_name` attribute must be used when the class inherits from more than one superclass to indicate the name of the superclass that contains the `component` element to be replaced by the contents of the `define` element.

A class in X-VRML can extend implementation of a superclass, in the sense that it can use the superclass implementation in the context of the subclass implementation. A code of the superclass implementation can be explicitly referenced by the use of the `super` element. The syntax of the `super` element is the following:

```
<super class_name="..." />
```

The element `super` is an empty element that is substituted by the processor with the actual implementation of the superclass. The `super` tag has one optional attribute: `class_name` that represents the name of the superclass the `super` element is referencing. If a class inherits only from one superclass, there is no need to specify the `class_name` attribute. In case of multiple inheritance, where there are a number of superclasses for the class, the `class_name` attribute is mandatory. There can be multiple references to a superclass in a class implementation. It means that the code of a superclass can be used multiple times in the subclass.

Some of the X-VRML classes may be translated by the X-VRML processor to VRML prototypes. X-VRML classes, however, are a much more powerful tool than VRML prototypes themselves. In particular, X-VRML classes provide inheritance, multiple inheritance, abstract classes, and the possibility of specifying arbitrary elements as attributes (e.g., other class names).

### 3.6.3 Creating Instances of Classes

Instances of the X-VRML classes can be created by the use of the `instance` element. The syntax of the `instance` element is the following:

```
<instance name="...">
  <attr name="..." value="..." />
  ...
</instance>
```

The `instance` tag has one mandatory attribute: `name`, which indicates the name of the X-VRML class. Only non-abstract classes may have instances. A class must be defined before its instances can be created. The `instance` element can contain only `attr` empty elements.

Each `attr` element defines a value of an attribute. Values of attributes that are not provided in the `instance` element are taken from the default values defined in the class definition.

### 3.7 X-VRML ARCO Module

#### 3.7.1 Overview

The X-VRML ARCO Module contains language elements specific to the ARCO project. These elements are used to enable efficient retrieval of various kinds of data from the ARCO database including folders, cultural objects, media objects, and AMS metadata.

Most of the functionality of the ARCO-specific X-VRML elements could be also achieved by the use of the Database and Core modules of X-VRML. However, use of the high-level elements has important advantages: it simplifies the code and makes the process of creating templates much less error-prone, and results in better performance due to reduction of the number of X-VRML elements that must be processed.

Properties of ARCO objects visualized in ARIF (folders, Cultural Objects, Media Objects, AMS) are retrieved by the use of a set of X-VRML commands. Instead of creating a specialized element for each property, elements permitting retrieval of all properties of a particular object type were created. The following elements are implemented in the second ARCO prototype:

- `<afProps>` for retrieval of ARIF Folder properties,
- `<coProps>` for retrieval of Cultural Object properties,
- `<moProps>` for retrieval of Media Object properties, and
- `<getAMS>` for retrieval of AMS metadata.

The X-VRML ARCO Module will be extended and refined as the system development continues.

#### 3.7.2 Retrieving Properties of ARIF Folders

To retrieve properties of ARIF Folders the `afProps` command is used. The `afProps` element has the following syntax:

```
<afProps afId="..." propName="..." varName="..." />
```

where:

- `afId` is the database identifier of the ARIF Folder,
- `propName` is a name of the folder property to be retrieved (see Table 2),
- `varName` is the name of a variable where the result is stored.

In Table 2, the list of valid `propName` attribute values is presented with description of the result produced by the `afProps` command.

PropName value	Result stored in the variable	X-VRML type of result
AF_NAME	The name of the ARIF folder	String
AF_DESCRIPTION	The description of the ARIF folder	String
PARENT_AF_ID	The database identifier of the parent folder	Integer

	folder	
CHILD_AF_IDS	A list of database identifiers of child folders (subfolders).	List
PARENT_AF_IDS	A list of database identifiers of all parent ARIF folders, forming a complete path to the folder identified by <code>afId</code> . The list is ordered from the parent folder id to the root folder id.	List
AF_PATH	A full path to the ARIF folder	String
CHILD_CO_IDS	A list of database identifiers of Cultural Objects assigned to the ARIF folder identified by <code>afId</code> .	List

**Table 2. Properties of ARIF Folders**

### 3.7.3 Retrieving Properties of Cultural Objects

To retrieve properties of Cultural Objects the `coProps` command is used. The `coProps` element has the following syntax:

```
<coProps coId="..." propName="..." varName="..." />
```

where:

- `coId` is a database identifier of the Cultural Object,
- `propName` is a name of the object property to be retrieved (see Table 3),
- `varName` is the name of a variable where the result is stored.

In Table 3, the list of valid `propName` attribute values is presented with description of the result produced by the `coProps` command.

PropName value	Result stored in the variable	X-VRML type of result
CHILD_MO_IDS	A list of database identifiers of Media Objects associated with the Cultural Object identified by <code>coId</code> .	List
CO_NAME	The name of the Cultural Object	String
CO_DESCRIPTION	The description of the Cultural Object	String
CO_TYPE	The type of the Cultural Object. May have two values: 'Acquired' and 'Refined'	String
REFINES_CO_ID	The database identifier of the parent Cultural Object in the refinement hierarchy if the object identified by <code>coId</code> is of type 'Refined', or NULL otherwise.	Integer
PARENT_AF_IDS	The list of database identifiers of the ARIF folders to which the Cultural	List

	Object identified by <code>coId</code> is assigned.	
--	---	--

**Table 3. Properties of Cultural Objects**

### 3.7.4 Retrieving Properties of Media Objects

To retrieve properties of Media Objects the `moProps` command is used. The `moProps` element has the following syntax:

```
<moProps moId="..." propName="..." varName="..." />
```

where:

- `moId` is a database identifier of the Media Object,
- `propName` is a name of the object property to be retrieved (see Table 4),
- `varName` is the name of a variable where the result is stored.

In Table 4, the list of valid `propName` attribute values is presented with description of the result produced by the `moProps` command.

PropName value	Result stored in variable	X-VRML type of result
MO_NAME	The name of the Media Object	String
MO_MIMETYPE	The MIME-type of the Media Object if any, or NULL value	String
MO_TYPE	The type of the Media Object Type of the Media Object, e.g. 'Simple Image', 'Panorama Image', 'Multi-resolution Image', '3Dstudio Max Project'	String
MO_DATA	The contents of the Media Object (e.g., a text value) if any, or NULL	String
MO_CREATION_DATE	The date of creation of the Media Object	String
PARENT_CO_IDS	The list of database identifiers of Cultural Objects to which the Media Object identified by <code>moId</code> is assigned.	List

**Table 4. Properties of Media Objects**

### 3.7.5 Retrieving AMS data

To retrieve formatted AMS metadata, the `getAMS` command is used. The `getAMS` element has the following syntax:

```
<getAMS type="..." id="..." xPath="..." xslId="..." varName="..." />
```

where:

- `type` represents type of the object (CO, MO, etc.),
- `id` is the object identifier in ARCO database,
- `xslId` is an XSL Stylesheet identifier in the ARCO database,
- `xPath` is an XPath expression which specifies the part of the AMS element to be retrieved,
- `varName` is an X-VRML variable name where the result of processing will be stored.

While interpreting the `getAMS` command, the X-VRML processor retrieves AMS data for the requested object. Using the XPath expression to extract the AMS element of interest, the X-VRML processor applies XSL stylesheet also retrieved from the database. The result is stored in a variable. Storing the result in a variable permits further processing of the data. It allows also to insert result into the result more than once without necessity to repeat the XSL stylesheet processing. The result of processing may be put in the resulting Web page by the use of the standard X-VRML `insert` command.

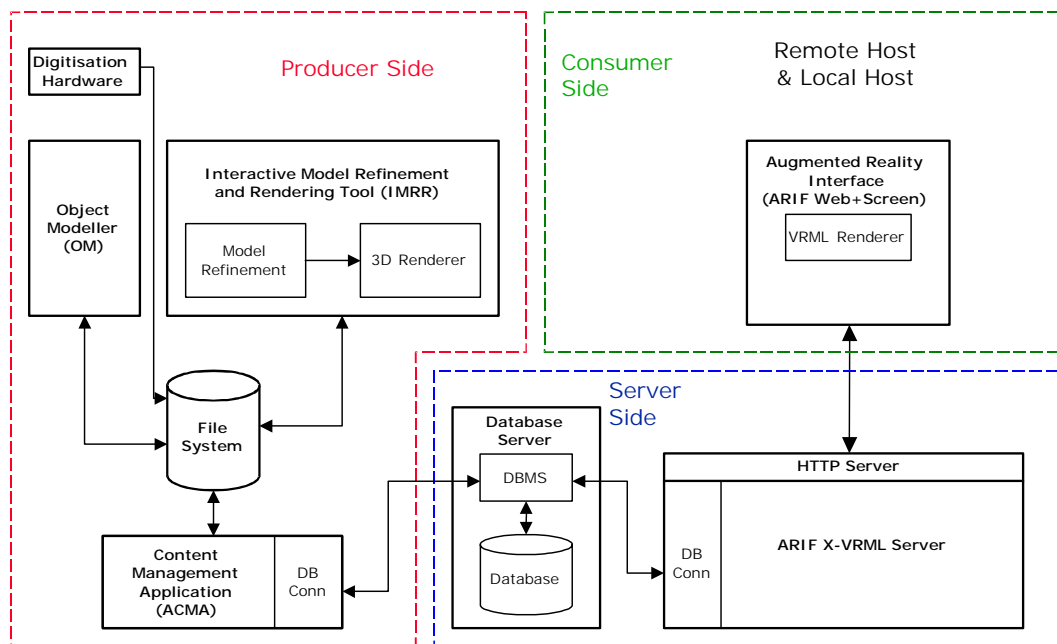


## 4. ARCO X-VRML Implementation

### 4.1 ARIF X-VRML Server

#### 4.1.1 Overview

The ARIF X-VRML Server is a software component responsible for generating content for augmented reality interfaces of the ARCO system. The content is dynamically generated based on data retrieved from the ARCO database and predefined X-VRML templates in response to user interaction. The position of the ARIF X-VRML Server module in the overall ARCO architecture is presented in Figure 87.



**Figure 87. ARIF X-VRML Server in the overall ARCO architecture**

The Web version of the ARIF Interface is accessed by the use of a standard HTML browser (such as Internet Explorer or Netscape Navigator) equipped with a VRML plug-in (such as Cosmo Player or ParallelGraphics Cortona). The communication between the browser and the server is performed by the use of the HTTP protocol. Standard HTTP server is employed on the ARCO server side.

The ARIF X-VRML Server processes requests generated by users, retrieves data from the ARCO database, combines the data and formats it accordingly to the XML-based X-VRML templates, and finally sends the generated output to the client browser. The output may contain both 2D HTML documents and 3D VRML models.

The content for the ARIF interface is generated dynamically based on:

- the X-VRML template(s),
- the data retrieved from the ARCO database,
- the user query,

- the user privileges,
- current state and configuration of the system.

The generated content consist of:

- 2D parts - HTML pages (possibly with interaction elements),
- 3D parts - VRML models of Cultural Objects and expositions (possibly with interaction elements), and
- embedded multimedia objects such as images, video and audio.

The X-VRML templates are used to generate both the 2D and 3D ARIF contents. In case of 2D parts the X-VRML templates generate HTML pages, in case of 3D parts the X-VRML templates generate VRML97 models – in future ARCO prototypes the X3D standard may be used. Both the 2D and 3D elements of ARIF may contain interaction elements. The interaction elements allow a user to specify a query or set preferences and submit these data to the ARIF X-VRML Server.

Both the 2D and 3D parts of ARIF may contain embedded multimedia objects, e.g. an HTML page may contain original images of an Acquired Object and a VRML model may contain the same or other images as textures.

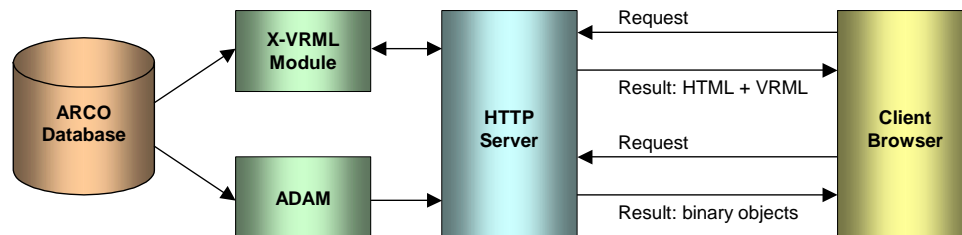
All elements that are displayed in ARIF are extracted by the ARIF X-VRML Server from the ARCO database. These include:

- 3D VRML models of Cultural Objects,
- multimedia objects (images, audio, video),
- textual descriptions (plain text and HTML),
- template graphics (backgrounds, bullets, buttons, etc.).

Also, the X-VRML templates and the configuration information are retrieved from the ARCO database.

#### 4.1.2 Architecture of the ARIF X-VRML Server

An overall architecture of the ARIF X-VRML Server is presented in Figure 88. The ARIF X-VRML Server consists of the X-VRML Module and the ADAM – ARCO Data Access Module subcomponents. These modules are described below.

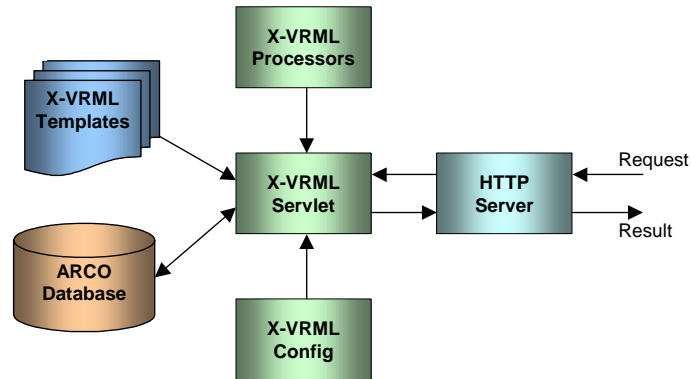


**Figure 88. Architecture of the ARIF X-VRML Server**

### 4.1.3 X-VRML Module

#### Overview of the X-VRML Module

The overall architecture of the X-VRML Module is presented in Figure 89.



**Figure 89. The X-VRML Module**

The X-VRML Module consists of the X-VRML Servlet, a set of X-VRML Processors, a set of X-VRML templates, and an HTTP server. The X-VRML Module is connected to the ARCO database.

The main part of the X-VRML Module is the X-VRML Servlet. The X-VRML Servlet is an interpretation engine of the X-VRML language. It is implemented in form of a servlet – a Java program being an extension to standard HTTP servers [19]. An important feature of the X-VRML Servlet is that implementation of the core servlet is independent of the implementation of particular X-VRML language elements. New elements implementing new features of the X-VRML language can be added to the system without modifying the main interpretation engine. Each element can be implemented as a separate Java class and added to the system. The X-VRML Servlet features also multi-level cache system, advanced database access module, remote configuration and monitoring, and powerful expression evaluator. The system can be used both with- and without a database.

The X-VRML servlet is connected to the HTTP server and processes requests with a specific URLs. The X-VRML Servlet reads X-VRML template files, processes them, and generates output in VRML or HTML depending on the template and the servlet configuration. Processing of particular X-VRML elements constituting a template is performed by *X-VRML Processors*. Processors are Java classes responsible for interpretation of the X-VRML elements. Mapping between the X-VRML elements and the classes that should be used to process these elements is provided in configuration files. Each element can be interpreted by a separate processor class or multiple elements can be processed by the same class. New processor classes can be added to the system by copying corresponding classes and updating the configuration information without the need to modify the X-VRML Servlet.

The most important features of the X-VRML Servlet are the following.

#### Generic execution engine

The X-VRML Servlet has a generic, language-independent execution engine. The system provides a framework for interpretation of models and execution of language-dependent X-VRML element processors. No modification to the servlet is required when new features are added to the X-VRML language.

#### API for preparing new X-VRML processors

All X-VRML processors are implemented as Java classes. The X-VRML Servlet provides API for preparing new processors. New elements can be added to the system by preparing new Java classes and modifying a configuration file.

#### XML-compliance

X-VRML language is an application of the XML standard. The model interpretation is performed by the use of standard XML tools. Moreover, correctness of X-VRML model syntax can be automatically verified.

#### Transparency

The X-VRML Servlet interprets only those XML tags that are used by the X-VRML language. All other tags are appropriately encoded before passing them to the XML parser. This allows use of the X-VRML Servlet and X-VRML language with output languages containing XML tags – such as X3D or HTML.

#### Caching

The X-VRML servlet is equipped with multilevel cache system. All results of processing are stored in the cache for later use.

#### Database access module

The X-VRML Servlet uses an advanced database connection interface. The interface allows to serve multiple simultaneous requests to the database in an efficient way and cache the results.

#### Customisation

Behaviour of the X-VRML Servlet can be easily controlled by a set of parameters. This allows adaptation of the system to different environments.

#### Access to X-VRML files by the HTTP protocol

X-VRML files used by the X-VRML Servlet are accessed by the use of the HTTP protocol. Therefore, the X-VRML files can be located on different hosts and maintained by different users. A user does not have to have access to the server to modify his/her X-VRML files.

In the second ARCO prototype the X-VRML Servlet accesses X-VRML meta-templates that retrieve the actual X-VRML templates from the ARCO database.

#### Expressions

X-VRML Servlet is equipped with an expression evaluator, enabling use of arithmetical, logical, and text operations on all command parameters.

#### Processor persistency

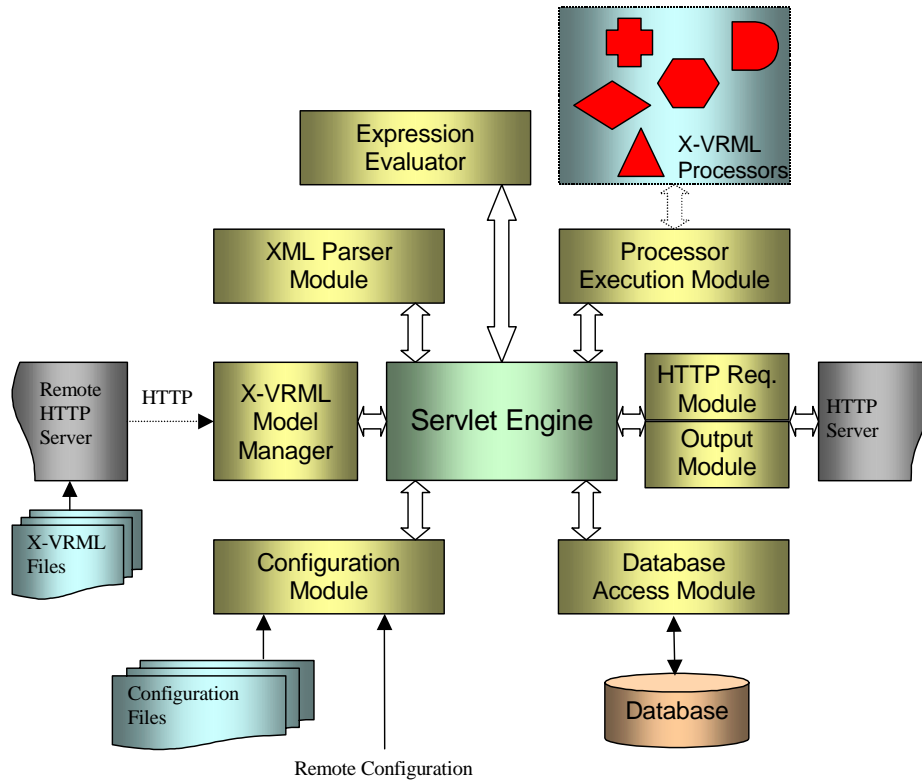
X-VRML processors can be used in two different ways by the X-VRML Servlet. Either a new instance of the processor class can be created for each element or the same instance can be used several times for all elements of the same type. Most of the elements are processed by the use of the first method. Some of the elements, however, may require execution history. In such case, the second method can be used.

#### Parameterisation

The X-VRML Servlet accepts parameters from different sources. These include configuration files, databases, and URLs. Accepting parameters from URLs enables easy implementation of interactive systems – a user may submit a query from a standard HTML page.

### Internal structure of the X-VRML Servlet

The internal structure of the X-VRML Servlet is presented in Figure 90.



**Figure 90. Internal structure of the X-VRML Servlet**

The main module of the X-VRML servlet is the Servlet Engine. This module organizes internal control in the servlet and enables communication between other servlet modules. The Engine uses eight other modules:

- X-VRML Model Manager,
- XML Parser,
- Expression Evaluator,
- Processor Execution Module,
- HTTP Request Module,
- Output Module,
- Database Access Module, and
- Configuration Module.

The X-VRML Model Manger is responsible for loading X-VRML files. The X-VRML files are loaded via the HTTP protocol from the same or a remote HTTP server. This increases system flexibility, in particular allows double processing of X-VRML models. An X-VRML model used by the X-VRML Servlet can be a result of previous processing of some meta-model.

The XML Parser Module is responsible for parsing the X-VRML file and building internal map of all XML elements that constitute the X-VRML model. The Expression Evaluator calculates values of all attributes prior to passing them to the Processor Execution Module.

The Processor Execution Module is responsible for invoking appropriate Processor classes to interpret particular XML elements. Mapping between XML elements and the Processor classes that should be used for interpretation is provided in the X-VRML Servlet configuration files.

The HTTP Request Module is responsible for handling requests from the HTTP server. The module identifies the request, retrieves values of parameters, and invokes the procedure of handing the request. The Output Module is responsible for gathering the output generated by Processor classes and sending it to the HTTP server.

The Database Access Module is responsible for maintaining connections to databases and caching results of database queries.

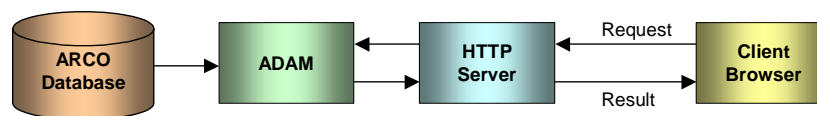
The Configuration Module reads configuration files and enables remote configuration and monitoring of the X-VRML Servlet by the use a standard HTML browser.

Two main configuration files are used by the X-VRML Servlet:

- X-VRML configuration file, which contains mapping between XML tags and corresponding Processors,
- Model configuration file, which contains a list of URL locations of collections of X-VRML models.

#### 4.1.4 ADAM – ARCO Data Access Module

All multimedia objects are delivered to the ARIF by the use of a special module – *ADAM – ARCO Data Access Module*. The ADAM module processes requests generated by client browsers in form of URLs, retrieves binary objects (Media Objects or Template Objects) from the ARCO database and delivers them back to the requesting client by the use of the HTTP protocol. All operations are performed in the streaming mode without saving the objects into local file system. The overall architecture of the ADAM subsystem is presented in Figure 91.



**Figure 91. Retrieving binary data by the ADAM subsystem**

The URLs processed by ADAM module contain database identifiers of the Media Objects or Template Objects that should be retrieved. The identifiers are generated by the use of X-VRML commands (cf. Section 3.7.3).

The ADAM component must return a valid MIME-type of the retrieved object. The identification of the MIME-type is performed by the ADAM module prior to object delivery.

The structure the URL for retrieving Media Objects is the following:

`http://dns/alias/getmo?id=nn`

The structure the URL for retrieving Template Objects is the following:

`http://dns/alias/getto?id=nn`

where

<code>http</code>	– is the specification of the communication protocol,
<code>dns</code>	– is the DNS name of the ARCO server,
<code>alias</code>	– is the alias to the ADAM servlet (depends on HTTP server configuration),
<code>nn</code>	– is the identifier of the binary object to be retrieved.

## 4.2 ARIF Dynamic Content

The X-VRML templates are used in the ARCO project to dynamically create content for the ARIF interface. The structure of the ARIF interface is defined by the structure of special ARIF Folders (also stored in the ARCO database). Each ARIF Folder may represent an exposition, a part of the exposition related to a particular subject, a museum room, etc. Subfolders may be used to divide exposition into smaller parts, e.g., focused on a particular topic.

The ARIF folders may contain three elements:

- Cultural Objects,
- Cultural Object Folders, and
- X-VRML Template Instances.

When an end user enters an ARIF Folder all Cultural Objects that are assigned to this particular ARIF Folder (either directly or indirectly by the use of the Cultural Object Folders) are displayed by the use of an *X-VRML Template Instance* that is assigned to this ARIF folder. The result is called *ARIF Presentation*.

The X-VRML Template Instance is an X-VRML template supplied with actual values for some of its formal parameters. The template parameter values are provided by a museum user (ARIF Content Designer) [4] by the use of a special *Presentation Manager* tool (cf. Section 4.4).

Depending on the set of parameters that are set in the X-VRML template instance, the end user (ARIF user) may be required (or not) to provide parameters for displaying the ARIF presentation contents. The following cases are possible:

- Some of the required template parameters are not set – the end user must first provide values for these parameters (e.g., search criteria) and then the ARIF presentation may be displayed,
- All required parameters are set but there are optional parameters that are not set – the ARIF presentation is displayed immediately, but the end user may change some of the presentation parameters (e.g. the default historical period) by the use of a special form,
- All template parameters are set – the ARIF presentation is displayed immediately and the user may not change its parameters.

This flexible assignment of attribute values for X-VRML templates make it possible to include in the ARIF interface search interfaces, customisable browse interfaces as well as fixed virtual expositions.

In order to speed-up the process of designing ARIF contents and ensure consistency of ARIF folder presentations the concept of inheritance of template instances was introduced. In this approach, if a specific ARIF folder does not contain its own template instance, the instance contained in its parent folder is taken by default (recursively). This solution enables using one template instance for the whole tree of folders in ARIF saving the preparation time and ensuring visual consistency of presentations.

A sample screen shot of the X-VRML Presentation Manager is presented in Figure 92. In the 'VR Museum' presentation an instance of the *Template1* is created, called '*ar2-2-03instance*'. This instance is valid for all subfolders of the 'VR Museum', namely: '*Ancient Greek room*', '*Modern room*', and '*Prehistoric room*', except '*Baroque room*'. Since the '*Ancient Greek room*' folder does not contain its own template instances, it uses the same visualization method as its parent folder. The '*Baroque room*' folder contains an instance of another template *Template2*, called '*ar2-1-02instance*'. Again, this instance is valid for this folder and for its subfolders: '*Angels in art*' and '*Cherubs in art*'.

Due to the X-VRML template parameterisation, different visualizations can be achieved by creation of template instances derived from the same template but supplied with different sets of parameter values. For example, the only difference between two instances of the same template in two folders may be a value of a parameter defining the background image.

To achieve maximum flexibility in organization of ARIF interfaces, the concept of ARIF domains was introduced. An ARIF domain is the environment in which the ARIF interface is used. For the second ARCO prototype two different domains are defined (Local Web and Remote Web), however, more domains may be defined in the later stages of the project.

Each X-VRML template is associated with an ARIF domain. In each ARIF folder multiple instances of templates for different domains may be created (but at most one for one ARIF domain). While accessing a presentation, a user agent has to specify which domain should be used. Then the appropriate instance of the template is used to produce the contents. Such organization permits to create different visualizations for different user agent environments, e.g., Remote, Local, etc.

The method of using X-VRML templates in ARIF, permits to create rich, parameterised visualizations, with respect to actual contents of the presentation and properties of the user agent.



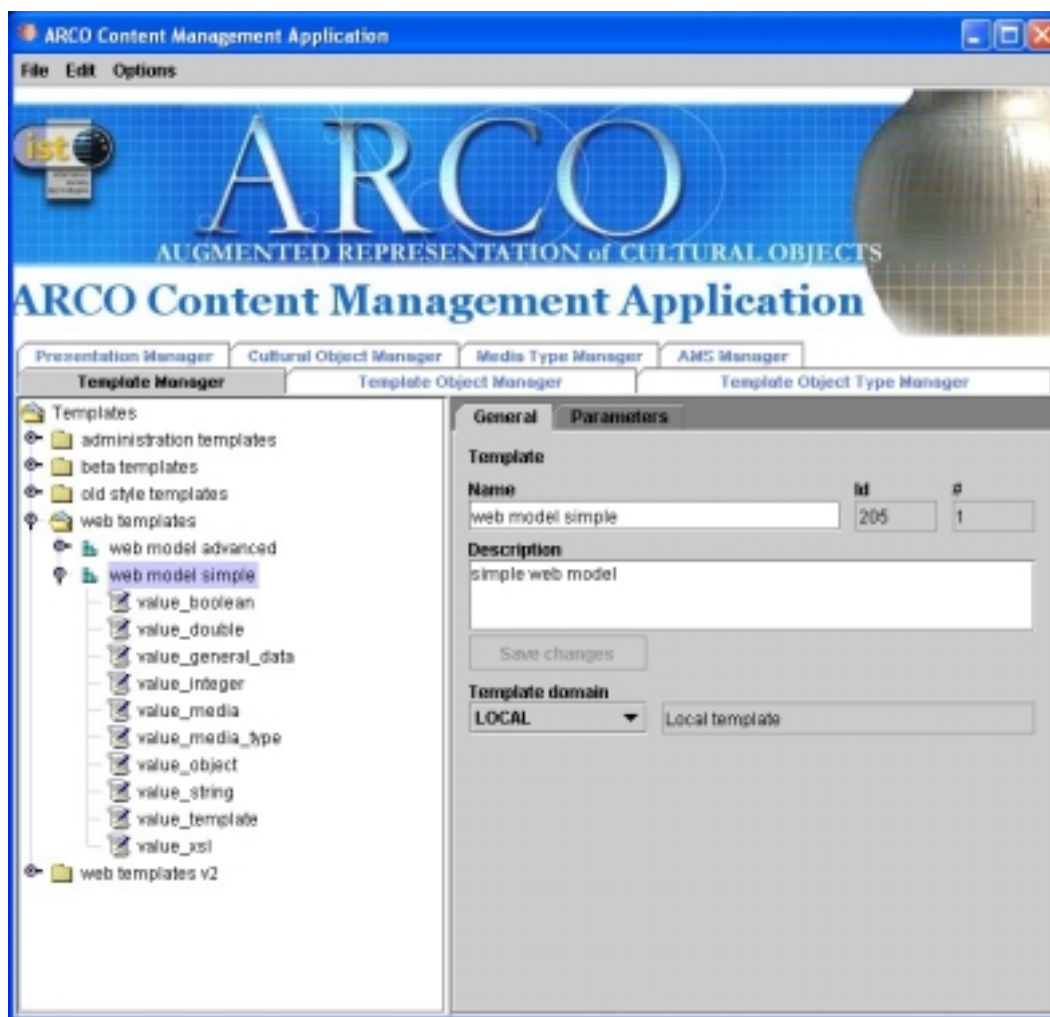


Figure 92. ARCO Presentation Manager

### 4.3 X-VRML Template Manager

The X-VRML Template Manager tool – integrated with ACMA – allows a user to administer X-VRML templates stored in the ARCO database. The X-VRML templates are stored in a hierarchical structure of template folders.

The Template Manager application is divided into two main panels. The left panel contains a tree that represents the structure of template folders, templates and their parameters. The tree is organized as follows. The root of the tree contains folders and templates not assigned to any folder. Each folder may contain other folders and templates. Each template may have a number of template parameters. There is an icon close to the name of each template. The icon represents the type of the X-VRML template. The main window of the ARCO Template Manager is presented in Figure 93.



**Figure 93. ARCO X-VRML Template Manager**

Detailed information about each item selected in the tree (such as the name, description, database ID, etc.) is displayed on the right panel. For folders and templates these are the name the description and the database ID. In case of a template parameter – the name, the label, the data type, the data type description, and the default value.

Additionally, for templates there is another tab panel (named “Parameters”) that contains information about parameters, their names, types and values in tabular form. Here the default values of template parameters can be set. The default value can be either a number or a string or a certain object (e.g. Media Object, Template Object). Double clicking on a value in parameter table causes a specialized editor to open. The parameters panel is presented in Figure 94.



**Figure 94. ARCO Template Manager – Template Parameters**

There is a popup menu available for each item on the tree where the most frequent actions are grouped. All Template Manager actions can be also accessed by the use of the main application menu.

#### **Folder actions**

The following actions are available for folders: creating (*Create folder*), deleting (*Delete folder*), and modifying its name and description. After modifications, a user can save changes into the database (*Save changes*).

#### **Template actions**

Templates can be loaded (*Load template*) from the disk by specifying the file containing the X-VRML model of the template. The X-VRML model of the template can be also retrieved from a database and saved to disk (*Save template*).

The X-VRML model can be edited from Template Manager by an external editor (*Edit X-VRML*). The default editor is assumed to be *Notepad* but user can set his own program in preferences (*Preferences*).

The name and description of each template can be modified. After modifications user can save changes into the database (*Save changes*).

The assignment of templates to folders in ARCO is a many-to-many relationship. This means that one template can be in more than one folder at the same time. This rule (which applies to all ARCO data) simplifies the organization of templates in the database. For example, all templates used by a museum can be put into one folder, at the same time all templates prepared by an author can be put into another folder. The facts of being used by a museum and being prepared by a particular author are independent.

Each template can be assigned (linked) to another folder (*Assign to another folder*) or moved (*Move*). It can also be removed from a folder (*Remove from folder*) or entirely deleted from all folders and the database (*Delete*). If a user tries to use the “remove from folder” function for the last instance of a template, a warning message is displayed and the user is prompted to use the delete function instead.

Templates can also be copied (*Copy*). Making a copy, unlike assigning, creates a duplicate of the template in the database.

#### 4.4 X-VRML Presentation Manager

The X-VRML Presentation Manager allows a user to administer ARIF contents by managing:

- the structure of ARIF Folders,
- X-VRML Template Instances, and
- Cultural Objects assigned to ARIF Folders.

The Presentation Manager application is divided into two main panels (Figure 92). The left panel contains a tree that presents the structure of ARIF folders, Template Instances and Cultural Objects. The tree is organized as follows. Each ARIF Folder can contain other ARIF Folders, Templates Instances, Cultural Objects and Cultural Object Folders. Each Template Instance can have number of embedded Template Instances. There is an icon close to the name of each object. The icon represents the type of the object.

A detailed information about each selected item in the tree is displayed on the right panel. For ARIF Folders, Cultural Objects and Template Instances these are the names, descriptions and database IDs. Additionally, for templates there is another tab-panel (named “Parameters”), which contains information about Template Instance parameters – their names, types and values. This panel allows to modify values of Template Instance parameters. A value can be either a number or a string or an object (e.g. Media Object or Template Object). Double clicking on a value in the parameter table opens a specialized editor.

There is a popup menu available for each item on the tree where the most frequent actions are grouped. All actions can be also accessed via the main application menu.

##### **ARIF Folder actions**

The following actions are available for ARIF Folders: creating (*Create*), deleting (*Delete*) and modifying name and description. After modifications a user can save changes into the database (*Save changes*).

Template instances can be created in ARIF Folder (*Create template instance*), deleted (*Delete*) or moved to another folder (*Move*).

Cultural objects and cultural object folders can be assigned to ARIF Folders (*Assign cultural object*, *Assign cultural object folder*), deleted (*Delete*) or moved (*Move*).

### **Template Instance actions**

The name and description of each Template Instance can be modified. After modifications a user can save changes into the database (*Save changes*).

## 5. XSL Stylesheets in ARCO

### 5.1 Overview

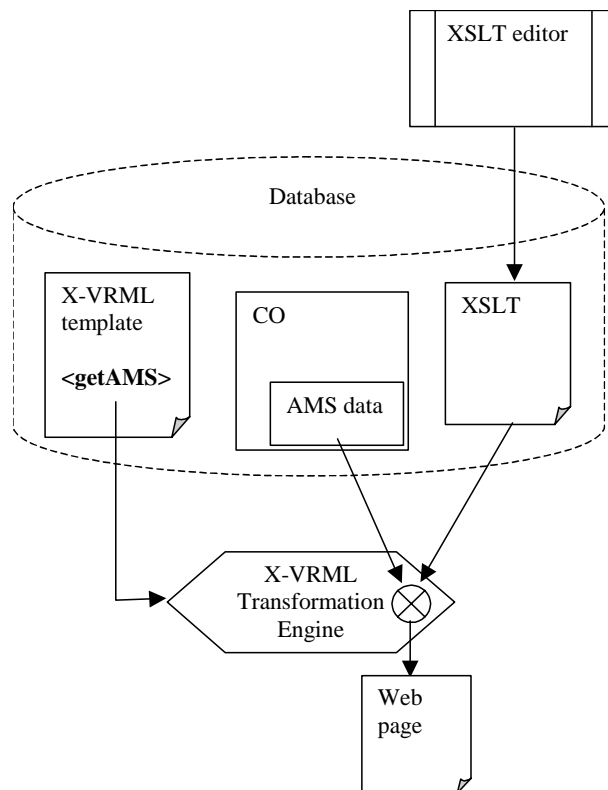
The XSL stylesheets are used within the ARCO system for displaying information stored in the ARCO database in the XML format. In the second ARCO prototype all AMS metadata descriptions of Cultural Objects and Media Objects are stored in the ARCO database in form of XML documents. The AMS metadata element sets are discussed in details in [1] and their XML implementation is described in Section 2.

The XSL stylesheets are used within the ARCO project in connection with the X-VRML templates, which are the base implementation technology for the ARIF Web Interfaces – *Local Web ARIF Interface* and *Remote Web ARIF Interface* [3]. The X-VRML language provides commands for retrieving XML data from the database and applying XSL stylesheets – also retrieved from the database (cf. Section 3.7.5).

The existence of advanced graphical user interfaces for designing XSL stylesheets makes the process of designing XSL template user friendly allowing the end-users (content designers) to generate their own XSL templates for displaying the AMS metadata in a customized way. No XSL programming experience is required to accomplish this task.

### 5.2 X-VRML – XSL Architecture

In Figure 95, the processing of X-VRML templates and XSL stylesheets in the second ARCO prototype is presented.



**Figure 95. Processing of AMS with X-VRML and XSLT**

The XSL stylesheets are used within the ARIF interfaces to present AMS metadata in a formatted human-readable way. The role of the XSL stylesheet is to select the elements of the AMS schema to be presented in the ARIF interface and provide their visualization properties (fonts, colours, etc.).

If the AMS metadata is displayed within the ARIF interface, the corresponding X-VRML template contains a specialized X-VRML element – `getAMS` responsible for processing of AMS with XSL (cf. Section 3.7.5). While interpreting the `getAMS` command, the X-VRML processor retrieves the AMS data for the requested object (Cultural Object or Media Object) from the ARCO database. Using the *XPath* expression to extract the AMS element of interest, the X-VRML processor applies XSL stylesheet – also retrieved from the database. The result of processing is temporarily stored in an X-VRML variable, and may be then put in the resulting Web page by the use of a standard X-VRML `insert` command.

The XSL template to be used and the *XPath* expression may be parameters of the X-VRML template providing maximum flexibility in displaying AMS in ARIF interfaces.

### 5.3 Designing XSL Templates for AMS Metadata

Existence of advanced graphical tools for designing XSL stylesheets allows users without XSL programming experience to design ARCO XSL stylesheets. The process of creating an XSL stylesheet for ARCO AMS metadata is presented in Figure 96. The designer must open the appropriate XML Schema (e.g., Cultural Object AMS). The list of available XML elements is displayed on the left side. On the right side, the format of the final HTML document can be designed. A user may add and format static page elements (e.g. labels) and include and format elements from the XML Schema. As a result, an XSLT stylesheet with appropriate element selection and formatting is generated. A sample XSLT stylesheet is presented in Appendix M.

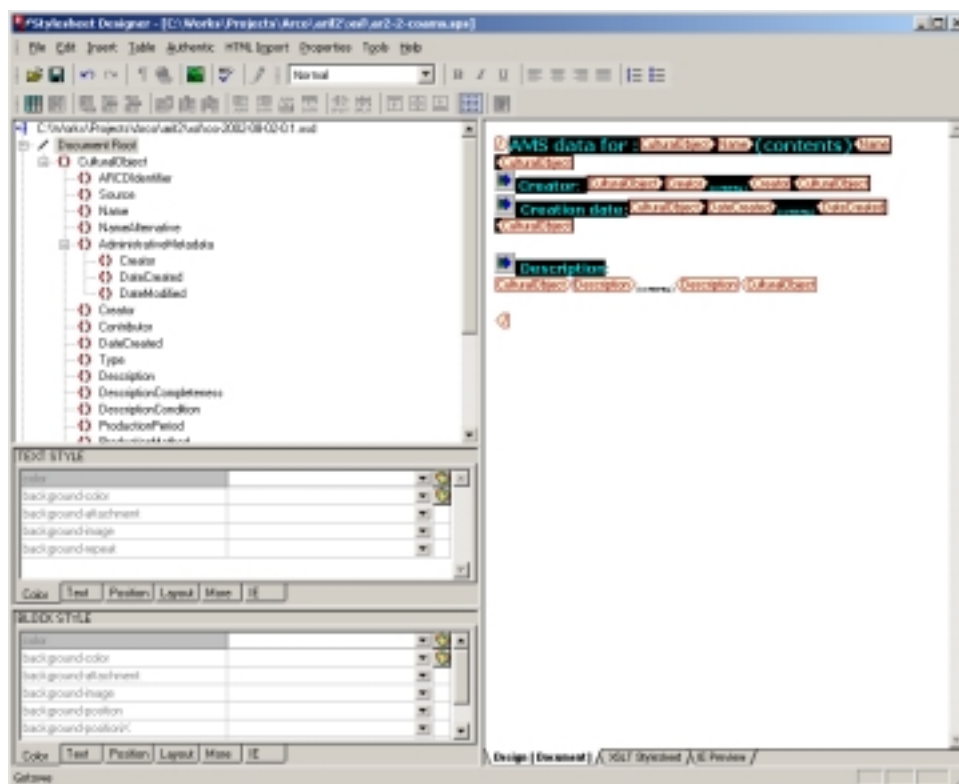
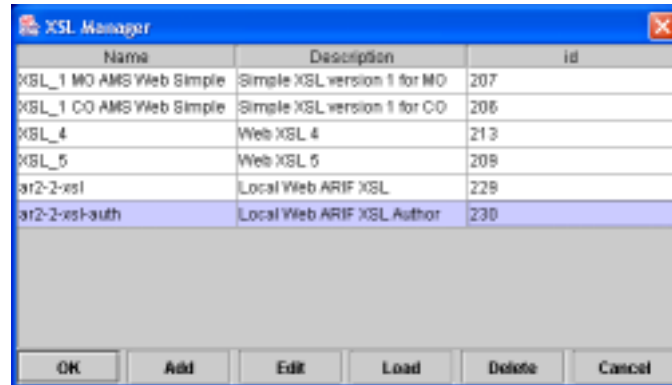


Figure 96. Designing AMS XSL template in Altova Stylesheet Designer

An XSL stylesheet prepared in such way may be stored in the ARCO database by the use of the XSL Template Manager available in ACMA (see Figure 97).



**Figure 97. ACMA XSL Manager**

In Figure 98, the result of processing Cultural Object AMS metadata with a sample XSL stylesheet is presented.





Figure 98. Cultural Object AMS metadata formatted with XSLT

## 6. References

- [1] ARCO Deliverable D8 – “Report on the XML descriptions of the database cultural objects”, ARCO Consortium 2002
- [2] ARCO Deliverable D3-SSP-ORDBMS – “Object Relational Database Management System”, Specification of the Second ARCO Prototype, ARCO Consortium 2002
- [3] ARCO Deliverable D3-SSP-ARIF – “Augmented Reality Interface”, Specification of the Second ARCO Prototype, ARCO Consortium 2002
- [4] ARCO Deliverable SSP-URS – “Second User Requirements Specification”, Specification of the Second ARCO Prototype, ARCO Consortium 2002
- [5] Rachel Heery and Manjula Patel, “Application profiles: mixing and matching metadata schemas”, *Ariadne*, Issue 25, 2000
- [6] Guidelines for implementing Dublin Core in XML, <http://www.dublincore.org/documents/2002/07/23/dc-xml-guidelines>
- [7] Recommendations for XML Schema for Qualified Dublin Core, <http://ukoln.ac.uk/metadata/dcmi/xmlschema/>
- [8] Database Working Group Charter, 1997; <http://www.vrml.org/WorkingGroups/dbwork/charter.html>
- [9] VRML Recommended Practices for SQL Database Access, Request for Proposals, 1997; <http://www.web3d.org/WorkingGroups/dbwork/dbrfp.html>
- [10] Lipkin, D., Database Extensions, Proposal for a VRML Informative Annex, Oracle Corporation, 1997; <http://www.web3d.org/WorkingGroups/dbwork/oracledbex.html>
- [11] Lipkin, D., Recommended Practices for SQL Database Access - Background and Overview, Oracle Corporation, <http://www.web3d.org/Recommended/vrml-sql/vrml-sql-ex.html>
- [12] Lipkin, D., Recommended Practices for SQL Database Access, VRML Informative Annex, 1998; <http://www.web3d.org/Recommended/vrml-sql/>
- [13] Walczak K., W. Cellary, XML-based Dynamic Generation of Virtual Scenes, Proc. of the 5th International Conference on Virtual Systems and Multimedia VSMM'99, pp. 335-348, Dundee, Scotland, UK, September 1-3, 1999
- [14] Walczak K., Database Modeling of Virtual Reality, Doctoral Dissertation, Technical University of Gdansk, Poland, 2001
- [15] Walczak K., W. Cellary, X-VRML – XML Based Modeling of Virtual Reality, Proc. of the 2002 International Symposium on Applications and the Internet (SAINT-2002), Nara, Japan, Jan. 28-Feb. 1, 2002, pp. 204-211
- [16] Walczak, K., W. Cellary, Building Database Applications of Virtual Reality with X-VRML, Proc. of the 7th International Conference on 3D Web Technology (Web3D-2002), Tempe, Arizona, USA, Feb. 24-28, 2002, pp. 111-120
- [17] X-VRML website, <http://xvrml.kti.ae.poznan.pl/>
- [18] XML - Extensible Markup Language, <http://www.w3.org/XML/>
- [19] Java Servlet Technology, The Power Behind the Server, <http://java.sun.com/products/servlet/>

## Appendix A. XML Schema for Administrative Metadata

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

<xsd:schema xmlns:arco="http://www.arco-web.org/namespaces/arco/v10"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xsd:annotation>

    <xsd:documentation>This schema was developed by the ARCO consortium for the 2nd
prototype of the ARCO project. Date 22-10-02. Developed by Jacek Chmielewski
jacek.chmielewski@kti.ae.poznan.pl. People involved: Krzysztof Walczak
walczak@kti.ae.poznan.pl</xsd:documentation>

  </xsd:annotation>

  <xsd:element name="administrativeMetadata">

    <xsd:annotation>

      <xsd:appinfo>

        <arco:displayName>Administrative Metadata</arco:displayName>

      </xsd:appinfo>

    </xsd:annotation>

    <xsd:complexType>

      <xsd:all>

        <xsd:element name="creator" type="xsd:string">

          <xsd:annotation>

            <xsd:appinfo>

              <arco:displayName>Creator</arco:displayName>

              <arco:autoGenerated>AdministrativeMedata_Creator</arco:autoGenerated>

            </xsd:appinfo>

          </xsd:annotation>

        </xsd:element>

        <xsd:element name="dateCreated" type="xsd:dateTime">

          <xsd:annotation>

            <xsd:appinfo>

              <arco:displayName>Date Created</arco:displayName>

              <arco:autoGenerated>AdministrativeMedata_DateCreated</arco:autoGenerated>

            </xsd:appinfo>

          </xsd:annotation>

        </xsd:element>

        <xsd:element name="dateModified" type="xsd:dateTime">

          <xsd:annotation>

            <xsd:appinfo>
```

```
<arco:displayName>Date Modified</arco:displayName>
<arco:autoGenerated>AdministrativeMetadata_DateModified</arco:autoGenerated>
</xsd:appinfo>
</xsd:annotation>
</xsd:element>
</xsd:all>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

## Appendix B. XML Schema for Cultural Object AMS

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

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:arco="http://www.arco-
web.org/namespaces/arco/v1.0" elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xsd:annotation>

    <xsd:documentation>This schema was developed by the ARCO consortium for the 2nd
    prototype of the ARCO project. Date 16-10-02. Developed by Nick Mourkoussis
    n.mourkoussis@sussex.ac.uk. People involved: Jacek Chmielewski
    jchmiel@kti.ae.poznan.pl, Krzysztof Walczak walczak@kti.ae.poznan.pl</xsd:documentation>

  </xsd:annotation>

  <xsd:element name="culturalObject">

    <xsd:annotation>

      <xsd:appinfo>

        <arco:displayName>Cultural Object</arco:displayName>

        <arco:definition>The cultural object Metadata Schema describes and structures
        curatorial information about the physical artefact. It is used by both Aquired and
        refined Objects</arco:definition>

      </xsd:appinfo>

    </xsd:annotation>

    <xsd:complexType>

      <xsd:sequence>

        <xsd:element name="source">

          <xsd:annotation>

            <xsd:appinfo>

              <arco:displayName>Source</arco:displayName>

              <arco:identifier>http://purl.org/dc/elements/1.1/source</arco:identifier>

              <arco:definition>A unique identifier for the physical artefact for which
              the CO is a surrogate</arco:definition>

              <arco:content>The identifier will be unique for the museum, e.g. LEWSA
              (institution).1973 (year when object was acquired for museum). 3 (a unique number for
              the donor). 5 (individual numbers for the objects within the donor's
              bequest)</arco:content>

            </xsd:appinfo>

          </xsd:annotation>

          <xsd:simpleType>

            <xsd:restriction base="xsd:string">

              <xsd:pattern value="\p{Lu}+:\d\d\d\d\d\.\d+\.\d+"/>

            </xsd:restriction>

          </xsd:simpleType>

        </xsd:element>

        <xsd:element name="name" type="xsd:string">
```

```

        <xsd:annotation>
        <xsd:appinfo>
            <arco:displayName>Name</arco:displayName>
            <arco:identifier>http://purl.org/dc/elements/1.1/title</arco:identifier>
            <arco:definition>A formal name given to the Cultural Object within the
ARCO system</arco:definition>
            <arco:content>Where possible, the name of the original artefact should be
used. Where a name does not exist, a caption could be used instead. Where a name or
caption does not exist, a one-sentence description of the content should be
added</arco:content>
        </xsd:appinfo>
    </xsd:annotation>
</xsd:element>

<xsd:element name="nameAlternative" type="xsd:string" minOccurs="0"
maxOccurs="unbounded">
    <xsd:annotation>
    <xsd:appinfo>
        <arco:displayName>Name Alternative</arco:displayName>
        <arco:identifier>http://purl.org/dc/terms/alternative</arco:identifier>
        <arco:definition>Any form of the name used as a substitute or alternative
to the formal name of the resource</arco:definition>
        <arco:content>alternative names to the artefact in textual
form</arco:content>
    </xsd:appinfo>
    </xsd:annotation>
</xsd:element>

<xsd:element name="creator" minOccurs="0" maxOccurs="unbounded">
    <xsd:annotation>
    <xsd:appinfo>
        <arco:displayName>Creator</arco:displayName>
        <arco:identifier>http://purl.org/dc/elements/1.1/creator</arco:identifier>
        <arco:definition>An entity primarily responsible for the creation of the
physical artefact</arco:definition>
        <arco:content>Personal names: the preferred form would provide last name
first, on general library indexing principles. Organisation names: organisation names
should be entered in direct order, the larger organisation first</arco:content>
    </xsd:appinfo>
    </xsd:annotation>
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="\p{Lu}\p{Ll}+, \s\p{Lu}\p{Ll}+" />
            <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*" />
        </xsd:restriction>
    </xsd:simpleType>

```

```

        </xsd:restriction>
    </xsd:simpleType>

</xsd:element>

<xsd:element name="contributor" minOccurs="0" maxOccurs="unbounded">
    <xsd:annotation>
        <xsd:appinfo>
            <arco:displayName>Contributor</arco:displayName>

            <arco:identifier>http://purl.org/dc/elements/1.1/contributor</arco:identifier>

            <arco:definition>An entity responsible for making contributions to the
creation of the physical artefact</arco:definition>

            <arco:content>This would be the name of the person or entity which
contributed to the creation of the cultural object, but is not the primary
creatorPersonal names: the preferred form would provide last name first, on general
library indexing principles. Organisation names: organisation names should be entered in
direct order, the larger organisation first</arco:content>
        </xsd:appinfo>
    </xsd:annotation>
</xsd:simpleType>
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="\p{Lu}\p{Ll}+, \s\p{Lu}\p{Ll}+" />
        <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*" />
    </xsd:restriction>
</xsd:simpleType>
</xsd:element>

<xsd:element name="dateCreated" minOccurs="0">
    <xsd:annotation>
        <xsd:appinfo>
            <arco:displayName>Date Created</arco:displayName>

            <arco:identifier>http://purl.org/dc/terms/created</arco:identifier>

            <arco:definition>Date of creation of the physical
artefact</arco:definition>

            <arco:content>Year or month.year or day.month.YYYY or
period</arco:content>
        </xsd:appinfo>
    </xsd:annotation>
</xsd:simpleType>
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="(\d\d\d\d)|(\d\d-\d\d\d\d)|(\d\d\-\d\d\d\d-
\d\d\d\d)|(\d\d\d\d\d=\d\d\d\d\d)|(\p{L}+((\s\p{L}+)|(\s\p{L}+))*" />
    </xsd:restriction>

```

```

    </xsd:simpleType>
  </xsd:element>

  <xsd:element name="type" type="xsd:string">

    <xsd:annotation>
      <xsd:appinfo>
        <arco:displayName>Type</arco:displayName>
        <arco:identifier>http://purl.org/dc/terms/type</arco:identifier>
        <arco:definition>The nature or genre of the cultural
object</arco:definition>
        <arco:content>SPECTRUM guidelines state that an object's name is 'a
description of the form or function of an object'. These are far too numerous to list
and different names will be necessary for different types of collections, which are
differentiated according to academic discipline eg, art, history, and archaeology.
Should be as clear and descriptive as possible e.g, jug, chair, print, sword, book. More
detail is then entered into under 'name alternative' e.g. book could then be described
as diary</arco:content>
      </xsd:appinfo>
    </xsd:annotation>
  </xsd:element>

  <xsd:element name="description" type="xsd:string">
    <xsd:annotation>
      <xsd:appinfo>
        <arco:displayName>Description</arco:displayName>
        <arco:identifier>http://purl.org/dc/terms/description</arco:identifier>
        <arco:definition>A short description characterising the physical
artefact</arco:definition>
        <arco:content>This should be a free text description of the physical
object. It should contain as much information as possible, as it will be the main
information that will be used by the end-user to judge its suitability for retrieval. It
should not assume too much expertise on behalf of the user, but should describe things
as specifically as possible</arco:content>
      </xsd:appinfo>
    </xsd:annotation>
  </xsd:element>

  <xsd:element name="completeness" minOccurs="0">
    <xsd:annotation>
      <xsd:appinfo>
        <arco:displayName>Completeness</arco:displayName>
        <arco:identifier>http://www.arco-
web.org/schemas/reach/completeness</arco:identifier>
        <arco:definition>The completeness of the object</arco:definition>
        <arco:content>One of Complete, Incomplete, Uncertain</arco:content>
      </xsd:appinfo>
    </xsd:annotation>
  </xsd:element>

```



```

<xsd:simpleType>
  <xsd:restriction base="xsd:string">

<xsd:enumeration value="Complete"/>

<xsd:enumeration value="Incomplete"/>
<xsd:enumeration value="Uncertain"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="condition" minOccurs="0">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:displayName>Condition</arco:displayName>
      <arco:identifier>http://www.arco-
web.org/schemas/reach/condition</arco:identifier>
      <arco:definition>A record of the condition of the physical
object</arco:definition>
      <arco:content>One of Poor, Fair, Good</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Poor"/>
      <xsd:enumeration value="Fair"/>
      <xsd:enumeration value="Good"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="productionPeriod" minOccurs="0">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:displayName>Production Period</arco:displayName>
      <arco:identifier>http://www.arco-
web.org/schemas/reach/period</arco:identifier>
      <arco:definition>A broader example of date when a stage in the design,
creation or manufacture of the object took place, which allows historical periods to be
expressed</arco:definition>
      <arco:content>This may be in textual form e.g., late 19th century, iOn
Age (using normal grammar and punctuation) or numerical form e.g., 1830=1860 (the = sign
is always sued to separate components). This is used for cataloguing and is recorded
only once</arco:content>
    </xsd:appinfo>
  </xsd:annotation>

```

```

</xsd:annotation>
<xsd:simpleType>
  <xsd:restriction base="xsd:string">
    <xsd:pattern
value="(\\d\\d\\d\\d=\\d\\d\\d\\d)|(\\p{L}+(\\s\\p{L}+)|(\\s\\p{L}+))*" />
  </xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="productionMethod" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:displayName>Production Method</arco:displayName>
      <arco:identifier>http://www.arco-
web.org/schemas/reach/method</arco:identifier>
      <arco:definition>This element may be found referred to as technique,
processes, methods, techniques and tools used to fabricate or decorate an
object</arco:definition>
      <arco:content>Single terms are used with no punctuation and terms for
specific tools that are sufficiently significant may be included</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>
<xsd:element name="formatMedium" maxOccurs="unbounded">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:displayName>Format Medium</arco:displayName>
      <arco:identifier>http://purl.org/dc/terms/medium</arco:identifier>
      <arco:definition>The substance(s) of which the object is
made</arco:definition>
      <arco:content>One of Amber, Tin, Brass, Wood Steel Metal (bell), Tin
(plated), Copper, Iron (wrought), Boxwood, Mahogany, Brass (cast), Lead, Bronze, Maple,
Ivory, Iron, Glass</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Amber" />
      <xsd:enumeration value="Boxwood" />
      <xsd:enumeration value="Brass" />
      <xsd:enumeration value="Brass (cast)" />
      <xsd:enumeration value="Bronze" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

```

```

        <xsd:enumeration value="Copper"/>
        <xsd:enumeration value="Glass"/>

        <xsd:enumeration value="Iron"/>
        <xsd:enumeration value="Iron (wrought)"/>

        <xsd:enumeration value="Ivory"/>

        <xsd:enumeration value="Lead"/>
        <xsd:enumeration value="Mahogany"/>
        <xsd:enumeration value="Maple"/>
        <xsd:enumeration value="Metal (bell)"/>
        <xsd:enumeration value="Steel"/>
        <xsd:enumeration value="Tin"/>
        <xsd:enumeration value="Tin (plated)"/>
        <xsd:enumeration value="Wood"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:element>

<xsd:element name="dimensions" minOccurs="0">
    <xsd:annotation>
        <xsd:appinfo>
            <arco:displayName>Dimensions</arco:displayName>
            <arco:identifier>http://purl.org/dc/terms//extent</arco:identifier>
            <arco:definition>Measurements associated with the three dimensions of the
object: height, width, depth</arco:definition>
            <arco:content>The dimensions should be given in the order height, width,
and depth. Use dimensional units appropriate are in millimetres</arco:content>
        </xsd:appinfo>
    </xsd:annotation>
</xsd:simpleType>
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="\d+\sx\s\d+\sx\s\d+"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:element>

<xsd:element name="coverageSpatial" minOccurs="0" maxOccurs="unbounded">
    <xsd:annotation>
        <xsd:appinfo>
            <arco:displayName>Coverage Spatial</arco:displayName>
            <arco:identifier>http://purl.org/dc/terms/spatial</arco:identifier>

```

```
<arco:definition>The geographical location in which an object was
created</arco:definition>

<arco:content>If a geographical locations is described with many regions.
regions should be delimited by commas</arco:content>
```

&lt;/xsd:appinfo&gt;

&lt;/xsd:annotation&gt;

```
<xsd:simpleType>
```

```
<xsd:restriction base="xsd:string">
```

```
<xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*"/>
```

&lt;/xsd:restriction&gt;

&lt;/xsd:simpleType&gt;

&lt;/xsd:element&gt;

```
<xsd:element name="components" minOccurs="0">
```

&lt;xsd:annotation&gt;

&lt;xsd:appinfo&gt;

```
<arco:displayName>Components</arco:displayName>
```

```
<arco:identifier>http://www.arco-  
web.org/schemas/amico/components</arco:identifier>
```

```
<arco:definition>The description of any parts/pieces of the work of  
art</arco:definition>
```

```
<arco:content>This component should be used when the physical artefact
consists more than one components. It consists of several "Component" sub-
elements</arco:content>
```

&lt;/xsd:appinfo&gt;

&lt;/xsd:annotation&gt;

&lt;xsd:complexType&gt;

&lt;xsd:sequence&gt;

```
<xsd:element name="component" type="xsd:string" maxOccurs="unbounded" />
```

&lt;/xsd:sequence&gt;

&lt;/xsd:complexType&gt;

&lt;/xsd:element&gt;

```
<xsd:element name="rights">
```

&lt;xsd:annotation&gt;

&lt;xsd:appinfo&gt;

```
<arco:displayName>Rights</arco:displayName>
```

```
<arco:identifier>http://purl.org/dc/elements/1.1/rights</arco:identifier>
```

```
<arco:definition>Information about rights held in and over the cultural  
artefact</arco:definition>
```

`<arco:content>`This would normally be a standard description of Cultural Object rights, or a link (URI) to a statement stored locally. A general statement could be assigned by the ARCO project`</arco:content>`

```

        </xsd:appinfo>
        </xsd:annotation>

        <xsd:simpleType>

            <xsd:restriction base="xsd:string"/>

        </xsd:simpleType>

    </xsd:element>

    <xsd:element name="owner" minOccurs="0">

        <xsd:annotation>

            <xsd:appinfo>

                <arco:displayName>Owner</arco:displayName>

                <arco:identifier>http://www.arco-
web.org/schemas/amico/owner</arco:identifier>

                <arco:definition>The name of the institution or individual who owns the
artefact</arco:definition>

                <arco:content>Organisation names: organisation names should be entered in
direct order, the larger organisation first. Personal names: the preferred form would
provide last name first, on general library indexing principles</arco:content>

            </xsd:appinfo>

        </xsd:annotation>

        <xsd:simpleType>

            <xsd:restriction base="xsd:string">

                <xsd:pattern value="\p{Lu}\p{Ll}+, \s\p{Lu}\p{Ll}+ " />

                <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*" />

            </xsd:restriction>

        </xsd:simpleType>

    </xsd:element>

</xsd:sequence>

</xsd:complexType>

</xsd:element>

</xsd:schema>

```

## Appendix C. XML Schema for Acquired Object AMS

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

<xsd:schema xmlns:arco="http://www.arco-web.org/namespaces/arco/v10"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <xsd:annotation>

    <xsd:documentation>This schema was developed by the ARCO consortium for the 2nd
  prototype of the ARCO project. Date 16-10-02. Developed by Nick Mourkoussis
  n.mourkoussis@sussex.ac.uk. People involved: Jacek Chmielewski
  jchmiel@kti.ae.poznan.pl, Krzysztof Walczak walczak@kti.ae.poznan.pl</xsd:documentation>

  </xsd:annotation>

  <xsd:element name="acquiredObject">

    <xsd:annotation>

      <xsd:appinfo>

        <arco:displayName>Acquired Object</arco:displayName>

        <arco:definition>The acquired object XML Schema describes the structure of AMS
  metadata associated with acquired objects</arco:definition>

      </xsd:appinfo>

    </xsd:annotation>

    <xsd:complexType>

      <xsd:sequence>

        <xsd:element name="identifier">

          <xsd:annotation>

            <xsd:appinfo>

              <arco:autoGenerated>AO_Identifier</arco:autoGenerated>

              <arco:displayName>Identifier</arco:displayName>

            </xsd:appinfo>

          </xsd:annotation>

          <arco:identifier>http://purl.org/dc/elements/1.1/identifier</arco:identifier>

          <arco:definition>A unique identifier assigned to a particular digital
  representation of the Cultural Object</arco:definition>

          <arco:content>Not Editable</arco:content>

        </xsd:appinfo>

      </xsd:annotation>

      <xsd:simpleType>

        <xsd:restriction base="xsd:string">

          <xsd:pattern value="\p{Lu}+\d+"/>

        </xsd:restriction>

      </xsd:simpleType>

    </xsd:element>

    <xsd:element name="name" type="xsd:string">
```

```

    <xsd:annotation>
    <xsd:appinfo>
        <arco:autoGenerated>AO_Name</arco:autoGenerated>
        <arco:displayName>Name</arco:displayName>
        <arco:identifier>http://purl.org/dc/elements/1.1/title</arco:identifier>
        <arco:definition>The name of the acquired object, or digital manifestation
of an artefact</arco:definition>
        <arco:content>This should be a name appropriate to a particular digital
representation of the Cultural Object and should distinguish it from other
representations or manifestations</arco:content>
    </xsd:appinfo>
</xsd:annotation>
</xsd:element>
<xsd:element name="publisher" minOccurs="0">
    <xsd:annotation>
    <xsd:appinfo>
        <arco:displayName>Publisher</arco:displayName>
        <arco:identifier>http://purl.org/dc/elements/1.1/publisher</arco:identifier>
        <arco:definition>An entity responsible for making the resource available
or accessible to others</arco:definition>
        <arco:content>Organisation names: organisation names should be entered in
direct order, the larger organisation first. Personal names: the preferred form would
provide last name first, on general library indexing principles</arco:content>
    </xsd:appinfo>
</xsd:annotation>
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*"/>
            <xsd:pattern value="\p{Lu}\p{Ll}+, \s\p{Lu}\p{Ll}+" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="creator">
    <xsd:annotation>
    <xsd:appinfo>
        <arco:autoGenerated>AO_Creator</arco:autoGenerated>
        <arco:displayName>Creator</arco:displayName>
        <arco:identifier>http://purl.org/dc/elements/1.1/creator</arco:identifier>
        <arco:definition>An entity primarily responsible for the creation of the
digital manifestation of the artefact</arco:definition>

```

```

        <arco:content>Not Editable</arco:content>
    </xsd:appinfo>

</xsd:annotation>
<xsd:simpleType>
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="contributor" minOccurs="0" maxOccurs="unbounded">
    <xsd:annotation>
        <xsd:appinfo>
            <arco:displayName>Contributor</arco:displayName>
            <arco:identifier>http://purl.org/dc/elements/1.1/contributor</arco:identifier>
            <arco:definition>An entity contributing to the creation of the digital
manifestation of the Cultural Object</arco:definition>
            <arco:content>Personal names: the preferred form would provide last name
first, on general library indexing principles Organisation names: organisation names
should be entered in direct order, the larger organisation first</arco:content>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="\p{Lu}\p{Ll}+,\s\p{Lu}\p{Ll}+"/>
            <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="dateCreated" type="xsd:dateTime">
    <xsd:annotation>
        <xsd:appinfo>
            <arco:autoGenerated>AO_DateCreated</arco:autoGenerated>
            <arco:displayName>Date Created</arco:displayName>
            <arco:identifier>http://purl.org/dc/terms/created</arco:identifier>
            <arco:definition>Date of creation of the Acquired Object</arco:definition>
            <arco:content>Not Editable</arco:content>
        </xsd:appinfo>
    </xsd:annotation>

```



```

</xsd:element>

<xsd:element name="description" type="xsd:string">

  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>AO_Description</arco:autoGenerated>
      <arco:displayName>Description</arco:displayName>
      <arco:identifier>http://purl.org/dc/elements/1.1/description</arco:identifier>
      <arco:definition>A short description characterising the digital
representation of the artefact</arco:definition>
      <arco:content>This should be a free text description of the acquired
object. It should contain as much information as possible, as it will be the main
information that will be used by the end-user to judge its suitability for retrieval. It
should not assume too much expertise on behalf of the user, but should describe things
as specifically as possible</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>

<xsd:element name="rights" type="xsd:string">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:displayName>Rights</arco:displayName>
      <arco:identifier>http://purl.org/dc/elements/1.1/rights</arco:identifier>
      <arco:definition>Information about rights held in and over the Refined
Object</arco:definition>
      <arco:content>This would normally be a standard description of AO rights,
e.g. "All rights are held by the ARCO Consortium" or a link (URI) to a statement stored
locally. A general statement could be assigned by the ARCO project</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>

<xsd:element name="format">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>AO_Format</arco:autoGenerated>
      <arco:displayName>Format</arco:displayName>
      <arco:identifier>http://purl.org/dc/elements/1.1/format</arco:identifier>
      <arco:definition>The format of an Refined Object</arco:definition>
      <arco:content>Not Editable</arco:content>
    </xsd:appinfo>
  </xsd:annotation>

```

```

<xsd:simpleType>
  <xsd:list>
</xsd:simpleType>

<xsd:simpleType>
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\P{L1}(\P{L}|\d|\+|\-|/|\.)+" />
  </xsd:restriction>
</xsd:simpleType>
</xsd:list>
</xsd:simpleType>
</xsd:element>
<xsd:element name="formatExtent">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>AO_FormatExtent</arco:autoGenerated>
      <arco:displayName>Format Extent</arco:displayName>
      <arco:identifier>http://purl.org/dc/terms/extent</arco:identifier>
      <arco:definition>The digital size of the Refined Object</arco:definition>
      <arco:content>Not Editable</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d+\s(B|KB|MB)" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>

```

## Appendix D. XML Schema for Refined Object AMS

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

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:arco="http://www.arco-
web.org/namespaces/arco/v1.0" elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xsd:annotation>

    <xsd:documentation>This schema was developed by the ARCO consortium for the 2nd
    prototype of the ARCO project. Date 22-10-02. Developed by Nick Mourkoussis
    n.mourkoussis@sussex.ac.uk. People involved: Jacek Chmielewski
    jchmiel@kti.ae.poznan.pl, Krzysztof Walczak walczak@kti.ae.poznan.pl</xsd:documentation>

  </xsd:annotation>

  <xsd:element name="refinedObject">

    <xsd:annotation>

      <xsd:appinfo>

        <arco:displayName>Refined Object</arco:displayName>

        <arco:definition>The refined object XML Schema describes the structure of AMS
        metadata associated with refined objects</arco:definition>

      </xsd:appinfo>

    </xsd:annotation>

    <xsd:complexType>

      <xsd:sequence>

        <xsd:element name="identifier">

          <xsd:annotation>

            <xsd:appinfo>

              <arco:autoGenerated>RO_Identifier</arco:autoGenerated>

              <arco:displayName>Identifier</arco:displayName>

            </xsd:appinfo>

          </xsd:annotation>

          <arco:identifier>http://purl.org/dc/elements/1.1/identifier</arco:identifier>

          <arco:definition>A unique identifier assigned to a particular digital
          representation of the Cultural Object</arco:definition>

          <arco:content>Not Editable</arco:content>

        </xsd:appinfo>

      </xsd:annotation>

      <xsd:simpleType>

        <xsd:restriction base="xsd:string">

          <xsd:pattern value="\p{Lu}+\d+"/>

        </xsd:restriction>

      </xsd:simpleType>

    </xsd:element>

    <xsd:element name="name" type="xsd:string">
```

```
<xsd:annotation>
  <xsd:appinfo>
    <arco:autoGenerated>RO_Name</arco:autoGenerated>
    <arco:displayName>Name</arco:displayName>
    <arco:identifier>http://purl.org/dc/elements/1.1/title</arco:identifier>
    <arco:definition>The name of the refined object, or digital manifestation
of an artefact</arco:definition>
    <arco:content>This should be a name appropriate to a particular digital
representation of the Cultural Object and should distinguish it from other
representations or manifestations</arco:content>
  </xsd:appinfo>
</xsd:annotation>
</xsd:element>
<xsd:element name="publisher" minOccurs="0">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:displayName>Publisher</arco:displayName>
      <arco:identifier>http://purl.org/dc/elements/1.1/publisher</arco:identifier>
      <arco:definition>An entity responsible for making the resource available
or accessible to others</arco:definition>
      <arco:content>Organisation names: organisation names should be entered in
direct order, the larger organisation first. Personal names: the preferred form would
provide last name first, on general library indexing principles</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*"/>
      <xsd:pattern value="\p{Lu}\p{Ll}+, \s\p{Lu}\p{Ll}+ "/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
<xsd:element name="creator">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>RO_Creator</arco:autoGenerated>
      <arco:displayName>Creator</arco:displayName>
      <arco:identifier>http://purl.org/dc/elements/1.1/creator</arco:identifier>
      <arco:definition>An entity primarily responsible for the creation of the
digital manifestation of the artefact</arco:definition>
```

```

        <arco:content>Not Editable</arco:content>
    </xsd:appinfo>

</xsd:annotation>
<xsd:simpleType>
    <xsd:restriction base="xsd:string">
        <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="contributor" minOccurs="0" maxOccurs="unbounded">
    <xsd:annotation>
        <xsd:appinfo>
            <arco:displayName>Contributor</arco:displayName>
            <arco:identifier>http://purl.org/dc/elements/1.1/contributor</arco:identifier>
            <arco:definition>An entity contributing to the creation of the digital
manifestation of the Cultural Object</arco:definition>
            <arco:content>Personal names: the preferred form would provide last name
first, on general library indexing principles Organisation names: organisation names
should be entered in direct order, the larger organisation first</arco:content>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="\p{Lu}\p{Ll}+,\s\p{Lu}\p{Ll}+"/>
            <xsd:pattern value="\p{L}+((\s\p{L}+)|(\s\p{L}+))*"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="dateCreated" type="xsd:dateTime">
    <xsd:annotation>
        <xsd:appinfo>
            <arco:autoGenerated>RO_DateCreated</arco:autoGenerated>
            <arco:displayName>Date Created</arco:displayName>
            <arco:identifier>http://purl.org/dc/terms/created</arco:identifier>
            <arco:definition>Date of creation of the refined Object</arco:definition>
            <arco:content>Not Editable</arco:content>
        </xsd:appinfo>
    </xsd:annotation>

```

```

</xsd:element>

<xsd:element name="description" type="xsd:string">

  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>RO_Description</arco:autoGenerated>
      <arco:displayName>Description</arco:displayName>
      <arco:identifier>http://purl.org/dc/elements/1.1/description</arco:identifier>
      <arco:definition>A short description characterising the digital
representation of the artefact</arco:definition>
      <arco:content>This should be a free text description of the refined
object. It should contain as much information as possible, as it will be the main
information that will be used by the end-user to judge its suitability for retrieval. It
should not assume too much expertise on behalf of the user, but should describe things
as specifically as possible</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>

<xsd:element name="rights" type="xsd:string">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:displayName>Rights</arco:displayName>
      <arco:identifier>http://purl.org/dc/elements/1.1/rights</arco:identifier>
      <arco:definition>Information about rights held in and over the Refined
Object</arco:definition>
      <arco:content>This would normally be a standard description of RO rights,
e.g. "All rights are held by the ARCO Consortium" or a link (URI) to a statement stored
locally. A general statement could be assigned by the ARCO project</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>

<xsd:element name="format">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>RO_Format</arco:autoGenerated>
      <arco:displayName>Format</arco:displayName>
      <arco:identifier>http://purl.org/dc/elements/1.1/format</arco:identifier>
      <arco:definition>The format of an Refined Object</arco:definition>
      <arco:content>Not Editable</arco:content>
    </xsd:appinfo>
  </xsd:annotation>

```

<pre> &lt;xsd:simpleType&gt;   &lt;xsd:list&gt; </pre>
<pre> &lt;xsd:simpleType&gt; </pre>
<pre>   &lt;xsd:restriction base="xsd:string"&gt;     &lt;xsd:pattern value="\P{Ll}(\P{L} \d \+ - / \.)+" /&gt;   &lt;/xsd:restriction&gt; &lt;/xsd:simpleType&gt; &lt;/xsd:list&gt; &lt;/xsd:simpleType&gt; &lt;/xsd:element&gt; &lt;xsd:element name="formatExtent"&gt;   &lt;xsd:annotation&gt;     &lt;xsd:appinfo&gt;       &lt;arco:autoGenerated&gt;RO_FormatExtent&lt;/arco:autoGenerated&gt;       &lt;arco:displayName&gt;Format Extent&lt;/arco:displayName&gt;       &lt;arco:identifier&gt;http://purl.org/dc/terms/extent&lt;/arco:identifier&gt;       &lt;arco:definition&gt;The digital size of the Refined Object&lt;/arco:definition&gt;       &lt;arco:content&gt;Not Editable&lt;/arco:content&gt;     &lt;/xsd:appinfo&gt;   &lt;/xsd:annotation&gt;   &lt;xsd:simpleType&gt;     &lt;xsd:restriction base="xsd:string"&gt;       &lt;xsd:pattern value="\d+\s(B KB MB)" /&gt;     &lt;/xsd:restriction&gt;   &lt;/xsd:simpleType&gt; &lt;/xsd:element&gt; &lt;xsd:element name="refines"&gt;   &lt;xsd:annotation&gt;     &lt;xsd:appinfo&gt;       &lt;arco:autoGenerated&gt;RO_Refines&lt;/arco:autoGenerated&gt;       &lt;arco:displayName&gt;Refines&lt;/arco:displayName&gt;       &lt;arco:identifier&gt;http://www.arco- web.org/schemas/arco/refines&lt;/arco:identifier&gt;       &lt;arco:definition&gt;The ordered list of identifiers (AO/Identifier, RO/Identifier) of Cultural Objects on the refinement path&lt;/arco:definition&gt;       &lt;arco:content&gt;Not Editable&lt;/arco:content&gt;     &lt;/xsd:appinfo&gt;   &lt;/xsd:annotation&gt; </pre>

```
<xsd:simpleType>
  <xsd:list>

<xsd:simpleType>

  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\p{Lu}+\d+"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:list>
</xsd:simpleType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```



## Appendix E. XML Schema for Media Object AMS

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

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:arco="http://www.arco-
web.org/namespaces/arco/v10" elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xsd:annotation>

    <xsd:documentation>This schema was developed by the ARCO consortium for the 2nd
    prototype of the ARCO project. Date 21-10-02. Developed by Jacek Chmielewski
    jchmiel@kti.ae.poznan.pl. People involved: Krzysztof Walczak walczak@kti.ae.poznan.pl,
    Nick Mourkoussis n.mourkoussis@sussex.ac.uk</xsd:documentation>

  </xsd:annotation>

  <xsd:element name="mediaObject">

    <xsd:complexType>

      <xsd:sequence>

        <xsd:element name="name" type="xsd:string">

          <xsd:annotation>

            <xsd:appinfo>

              <arco:autoGenerated>MO_Name</arco:autoGenerated>

              <arco:displayName>Name</arco:displayName>

              <arco:identifier>http://purl.org/dc/elements/1.1/title</arco:identifier>

              <arco:definition>A name given to the Media Object</arco:definition>

              <arco:content>This should be a name appropriate to a particular digital
              representation of the Cultural Object and should distinguish it from other
              representations or manifestations</arco:content>

            </xsd:appinfo>

          </xsd:annotation>

        </xsd:element>

        <xsd:element name="type">

          <xsd:annotation>

            <xsd:appinfo>

              <arco:autoGenerated>MO_Type</arco:autoGenerated>

              <arco:displayName>Type</arco:displayName>

              <arco:identifier>http://purl.org/dc/elements/1.1/type</arco:identifier>

              <arco:definition>The type of a Media Object</arco:definition>

              <arco:content>Not Editable</arco:content>

            </xsd:appinfo>

          </xsd:annotation>

          <xsd:simpleType>

            <xsd:list>

              <xsd:simpleType>
```

```

        <xsd:restriction base="xsd:string">
            <xsd:pattern value="\P{Ll}(\P{L}|\d|\+|\-|/|\.)+" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:list>
</xsd:simpleType>
</xsd:element>
<xsd:element name="subject" minOccurs="0">
    <xsd:annotation>
        <xsd:appinfo>
            <arco:displayName>Subject</arco:displayName>
            <arco:identifier>http://purl.org/dc/elements/1.1/subject</arco:identifier>
            <arco:definition>Keywords which identify the subjects of the contents of
the digital manifestation</arco:definition>
            <arco:content>Keywords should be delimited by spaces</arco:content>
        </xsd:appinfo>
    </xsd:annotation>
</xsd:simpleType>
<xsd:list>
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="\P{Lu}\P{Ll}+" />
        </xsd:restriction>
    </xsd:simpleType>
</xsd:list>
</xsd:simpleType>
</xsd:element>
<xsd:element name="description" type="xsd:string" minOccurs="0">
    <xsd:annotation>
        <xsd:appinfo>
            <arco:displayName>Description</arco:displayName>
            <arco:identifier>http://purl.org/dc/elements/1.1/description</arco:identifier>
            <arco:definition>A short description characterising the subject content of
the digital manifestation</arco:definition>
            <arco:content>This should be a free text description. It should not assume
too much expertise on behalf of the user, but should describe things as specifically as
possible</arco:content>
        </xsd:appinfo>
    </xsd:annotation>

```

```
</xsd:element>

<xsd:element name="dateCreated" type="xsd:dateTime">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>MO_DateCreated</arco:autoGenerated>
      <arco:displayName>Date Created</arco:displayName>
      <arco:identifier>http://purl.org/dc/terms /created</arco:identifier>
      <arco:definition>Date of creation of the digital
manifestation</arco:definition>
      <arco:content>Not Editable</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>

<xsd:element name="creator">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>MO_Creator</arco:autoGenerated>
      <arco:displayName>Creator</arco:displayName>
      <arco:identifier>http://purl.org/dc/elements/1.1/creator</arco:identifier>
      <arco:definition>An entity primarily responsible for the creation of the
digital manifestation</arco:definition>
      <arco:content>Not Editable</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\P{Lu}\P{Ll}+(\, \s\P{Lu}\P{Ll}+)*"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="formatExtent">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>MO_FormatExtent</arco:autoGenerated>
      <arco:displayName>Format Extent</arco:displayName>
      <arco:identifier>http://purl.org/dc/terms/extent</arco:identifier>
      <arco:definition>The size of the digital manifestation</arco:definition>
      <arco:content>Not Editable</arco:content>
    </xsd:appinfo>
  </xsd:annotation>

```

```
</xsd:appinfo>
</xsd:annotation>
<xsd:simpleType>
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d+\s(B|KB|MB)"/>
  </xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="rights" type="xsd:string">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:displayName>Rights</arco:displayName>
      <arco:identifier>http://purl.org/dc/elements/1.1/rights</arco:identifier>
      <arco:definition>Information about rights held in and over the digital
manifestation</arco:definition>
      <arco:content>This would contain either a free-text rights management
statement for the artefact or a URI reference to such a statement</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

## Appendix F. XML Schema for Media Object Simple Image AMS

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

<xsd:schema xmlns:arco="http://www.arco-web.org/namespaces/arco/v10"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xsd:annotation>

    <xsd:documentation>This schema was developed by the ARCO consortium for the 2nd
prototype of the ARCO project. Date 22-10-02. Developed by Nick Mourkoussis
n.mourkoussis@sussex.ac.uk. People involved: Jacek Chmielewski
jchmiel@kti.ae.poznan.pl, Krzysztof Walczak walczak@kti.ae.poznan.pl</xsd:documentation>

  </xsd:annotation>

  <xsd:element name="mediaObjectSimpleImage">

    <xsd:annotation>

      <xsd:appinfo>

        <arco:displayName>Simple Image Media Object</arco:displayName>

        <arco:definition>AMS schema described by metadata associated with simple image
media objects</arco:definition>

      </xsd:appinfo>

    </xsd:annotation>

    <xsd:complexType>

      <xsd:sequence>

        <xsd:element name="technique" maxOccurs="unbounded">

          <xsd:annotation>

            <xsd:appinfo>

              <arco:displayName>Technique</arco:displayName>

              <arco:identifier>http://www.arco-
web.org/schemas/arco/technique</arco:identifier>

              <arco:definition>The technique used to acquire an image of the Cultural
Object</arco:definition>

              <arco:content>This should be one item from predefined dictionary.Digital
photography, Scanning, Manual Painting</arco:content>

            </xsd:appinfo>

          </xsd:annotation>

          <xsd:simpleType>

            <xsd:restriction base="xsd:string">

              <xsd:enumeration value="Digital Photography"/>

              <xsd:enumeration value="Manual Painting"/>

              <xsd:enumeration value="Scanning"/>

            </xsd:restriction>

          </xsd:simpleType>

        </xsd:element>

      </xsd:sequence>

    </xsd:complexType>

  </xsd:element>

</xsd:schema>
```

```
</xsd:element>

<xsd:element name="imageSize">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>MOT_SimpleImage_ImageSize</arco:autoGenerated>
      <arco:displayName>Image Size</arco:displayName>
      <arco:identifier>http://www.arco-
web.org/schemas/arco/size</arco:identifier>
      <arco:definition>Image width and height in pixels</arco:definition>
      <arco:content>Not Editable</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d+x\d+"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="resolution" type="xsd:string">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>MOT_SimpleImage_Resolution</arco:autoGenerated>
      <arco:displayName>Resolution</arco:displayName>
      <arco:identifier>http://www.arco-
web.org/schemas/arco/resolution</arco:identifier>
      <arco:definition>Image resolution in dpi</arco:definition>
      <arco:content>Not Editable</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>

<xsd:element name="compressionMethod">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>MOT_SimpleImage_CompressionMethod</arco:autoGenerated>
      <arco:displayName>Compression method</arco:displayName>
      <arco:identifier>http://www.arco-web.org/schemas/arco/compression-
method</arco:identifier>
      <arco:definition>Compression method used to create this image
file</arco:definition>
```

```
<arco:content>Not Editable</arco:content>

</xsd:appinfo>

</xsd:annotation>

<xsd:simpleType>

  <xsd:restriction base="xsd:string">

    <xsd:enumeration value="raw"/>

    <xsd:enumeration value="jpeg"/>

    <xsd:enumeration value="gif"/>

  </xsd:restriction>

</xsd:simpleType>

</xsd:element>

<xsd:element name="compressionFactor">

  <xsd:annotation>

    <xsd:appinfo>

      <arco:displayName>Compression factor</arco:displayName>

      <arco:identifier>http://www.arco-web.org/schemas/arco/compression-
factor</arco:identifier>

      <arco:definition>Compression algorithm strength factor</arco:definition>

      <arco:content>One item from predefined dictionaryNone, High, Medium,
Low</arco:content>

    </xsd:appinfo>

  </xsd:annotation>

  <xsd:simpleType>

    <xsd:restriction base="xsd:string">

      <xsd:enumeration value="High"/>

      <xsd:enumeration value="Low"/>

      <xsd:enumeration value="Medium"/>

      <xsd:enumeration value="None"/>

    </xsd:restriction>

  </xsd:simpleType>

</xsd:element>

<xsd:element name="colourDepth" type="xsd:integer">

  <xsd:annotation>

    <xsd:appinfo>

      <arco:autoGenerated>MOT_SimpleImage_ColourDepth</arco:autoGenerated>

      <arco:displayName>Colour Depth</arco:displayName>

      <arco:identifier>http://www.arco-web.org/schemas/arco/colour-
depth</arco:identifier>
```

```
        <arco:definition>Number of bits that represents single colour  
point</arco:definition>  
        <arco:content>Not Editable</arco:content>  
    </xsd:appinfo>  
    </xsd:annotation>  
    </xsd:element>  
    </xsd:sequence>  
    </xsd:complexType>  
    </xsd:element>  
</xsd:schema>
```



## Appendix G. XML Schema for Media Object Description AMS

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

<xsd:schema xmlns:arco="http://www.arco-web.org/namespaces/arco/v10"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xsd:annotation>

    <xsd:documentation>This schema was developed by the ARCO consortium for the 2nd
prototype of the ARCO project. Date 22-10-02. Developed by Nick Mourkoussis
n.mourkoussis@sussex.ac.uk. People involved: Jacek Chmielewski
jchmiel@kti.ae.poznan.pl, Krzysztof Walczak walczak@kti.ae.poznan.pl</xsd:documentation>

  </xsd:annotation>

  <xsd:element name="mediaObjectDescription">

    <xsd:annotation>

      <xsd:appinfo>

        <arco:displayName>Description Media Object</arco:displayName>

        <arco:definition>AMS schema described by metadata associated with the
description media objects</arco:definition>

      </xsd:appinfo>

    </xsd:annotation>

    <xsd:complexType>

      <xsd:sequence>

        <xsd:element name="technique" maxOccurs="unbounded">

          <xsd:annotation>

            <xsd:appinfo>

              <arco:displayName>Technique</arco:displayName>

              <arco:identifier>http://www.arco-
web.org/schemas/arco/technique</arco:identifier>

              <arco:definition>The technique used to acquire a description of the
Cultural Object</arco:definition>

              <arco:content>This should be one item from predefined dictionary. Typing in
an editing tool, OCR</arco:content>

            </xsd:appinfo>

          </xsd:annotation>

          <xsd:simpleType>

            <xsd:restriction base="xsd:string">

              <xsd:enumeration value="OCR"/>

              <xsd:enumeration value="Typing in an editing tool"/>

            </xsd:restriction>

          </xsd:simpleType>

        </xsd:element>

      </xsd:sequence>

    </xsd:complexType>

  </xsd:element>

</xsd:element>
```

```
<xsd:element name="length" type="xsd:long">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>MOT_Description_Length</arco:autoGenerated>
      <arco:displayName>Length</arco:displayName>
      <arco:identifier>http://www.arco-
web.org/schemas/arco/length</arco:identifier>
      <arco:definition>Text length in chars</arco:definition>
      <arco:content>Not Editable</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>

<xsd:element name="characterSet" minOccurs="0">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:displayName>Character set</arco:displayName>
      <arco:identifier>http://www.arco-web.org/schemas/arco/character-
set</arco:identifier>
      <arco:definition>Character set used in description text</arco:definition>
      <arco:content>One Identifier of character set defined on the
http://www.iana.org/assignments/character-sets</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="ANSI_X3.4-1968"/>
      <xsd:enumeration value="ISO-10646-UTF-1"/>
      <xsd:enumeration value="INVARIANT"/>
      <xsd:enumeration value="ISO_646.irv:1983"/>
      <xsd:enumeration value="BS_4730"/>
      <xsd:enumeration value="NATS-SEFI"/>
      <xsd:enumeration value="NATS-SEFI-ADD"/>
      <xsd:enumeration value="NATS-DANO"/>
      <xsd:enumeration value="NATS-DANO-ADD"/>
      <xsd:enumeration value="SEN_850200_B"/>
      <xsd:enumeration value="SEN_850200_C"/>
      <xsd:enumeration value="KS_C_5601-1987"/>
      <xsd:enumeration value="ISO-2022-KR"/>
      <xsd:enumeration value="EUC-KR"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

```
<xsd:enumeration value="ISO-2022-JP" />
<xsd:enumeration value="ISO-2022-JP-2" />
<xsd:enumeration value="ISO-2022-CN" />
<xsd:enumeration value="ISO-2022-CN-EXT" />
<xsd:enumeration value="JIS_C6220-1969-jp" />
<xsd:enumeration value="JIS_C6220-1969-ro" />
<xsd:enumeration value="IT" />
<xsd:enumeration value="PT" />
<xsd:enumeration value="ES" />
<xsd:enumeration value="greek7-old" />
<xsd:enumeration value="latin-greek" />
<xsd:enumeration value="DIN_66003" />
<xsd:enumeration value="NF_Z_62-010_(1973)" />
<xsd:enumeration value="Latin-greek-1" />
<xsd:enumeration value="ISO_5427" />
<xsd:enumeration value="JIS_C6226-1978" />
<xsd:enumeration value="BS_viewdata" />
<xsd:enumeration value="INIS" />
<xsd:enumeration value="INIS-8" />
<xsd:enumeration value="INIS-cyrillic" />
<xsd:enumeration value="ISO_5427:1981" />
<xsd:enumeration value="ISO_5428:1980" />
<xsd:enumeration value="GB_1988-80" />
<xsd:enumeration value="GB_2312-80" />
<xsd:enumeration value="NS_4551-1" />
<xsd:enumeration value="NS_4551-2" />
<xsd:enumeration value="NF_Z_62-010" />
<xsd:enumeration value="videotex-suppl" />
<xsd:enumeration value="PT2" />
<xsd:enumeration value="ES2" />
<xsd:enumeration value="MSZ_7795.3" />
<xsd:enumeration value="JIS_C6226-1983" />
<xsd:enumeration value="greek7" />
<xsd:enumeration value="ASMO_449" />
<xsd:enumeration value="iso-ir-90" />
<xsd:enumeration value="JIS_C6229-1984-a" />
<xsd:enumeration value="JIS_C6229-1984-b" />
```

```
<xsd:enumeration value="JIS_C6229-1984-b-add" />
<xsd:enumeration value="JIS_C6229-1984-hand" />
<xsd:enumeration value="JIS_C6229-1984-hand-add" />
<xsd:enumeration value="JIS_C6229-1984-kana" />
<xsd:enumeration value="ISO_2033-1983" />
<xsd:enumeration value="ANSI_X3.110-1983" />
<xsd:enumeration value="ISO_8859-1:1987" />
<xsd:enumeration value="ISO_8859-2:1987" />
<xsd:enumeration value="T.61-7bit" />
<xsd:enumeration value="T.61-8bit" />
<xsd:enumeration value="ISO_8859-3:1988" />
<xsd:enumeration value="ISO_8859-4:1988" />
<xsd:enumeration value="ECMA-cyrillic" />
<xsd:enumeration value="CSA_Z243.4-1985-1" />
<xsd:enumeration value="CSA_Z243.4-1985-2" />
<xsd:enumeration value="CSA_Z243.4-1985-gr" />
<xsd:enumeration value="ISO_8859-6:1987" />
<xsd:enumeration value="ISO_8859-6-E" />
<xsd:enumeration value="ISO_8859-6-I" />
<xsd:enumeration value="ISO_8859-7:1987" />
<xsd:enumeration value="T.101-G2" />
<xsd:enumeration value="ISO_8859-8:1988" />
<xsd:enumeration value="ISO_8859-8-E" />
<xsd:enumeration value="ISO_8859-8-I" />
<xsd:enumeration value="CSN_369103" />
<xsd:enumeration value="JUS_I.B1.002" />
<xsd:enumeration value="ISO_6937-2-add" />
<xsd:enumeration value="IEC_P27-1" />
<xsd:enumeration value="ISO_8859-5:1988" />
<xsd:enumeration value="JUS_I.B1.003-serb" />
<xsd:enumeration value="JUS_I.B1.003-mac" />
<xsd:enumeration value="ISO_8859-9:1989" />
<xsd:enumeration value="greek-ccitt" />
<xsd:enumeration value="NC_NC00-10:81" />
<xsd:enumeration value="ISO_6937-2-25" />
<xsd:enumeration value="GOST_19768-74" />
<xsd:enumeration value="ISO_8859-suppl" />
```

```
<xsd:enumeration value="ISO_10367-box"/>
<xsd:enumeration value="ISO-8859-10"/>
<xsd:enumeration value="latin-lap"/>
<xsd:enumeration value="JIS_X0212-1990"/>
<xsd:enumeration value="DS_2089"/>
<xsd:enumeration value="us-dk"/>
<xsd:enumeration value="dk-us"/>
<xsd:enumeration value="JIS_X0201"/>
<xsd:enumeration value="ISO-10646-UCS-2"/>
<xsd:enumeration value="DEC-MCS"/>
<xsd:enumeration value="hp-roman8"/>
<xsd:enumeration value="macintosh"/>
<xsd:enumeration value="IBM037"/>
<xsd:enumeration value="IBM038"/>
<xsd:enumeration value="IBM273"/>
<xsd:enumeration value="IBM274"/>
<xsd:enumeration value="IBM275"/>
<xsd:enumeration value="IBM278"/>
<xsd:enumeration value="IBM280"/>
<xsd:enumeration value="IBM281"/>
<xsd:enumeration value="IBM284"/>
<xsd:enumeration value="IBM285"/>
<xsd:enumeration value="IBM290"/>
<xsd:enumeration value="IBM297"/>
<xsd:enumeration value="IBM420"/>
<xsd:enumeration value="IBM423"/>
<xsd:enumeration value="IBM424"/>
<xsd:enumeration value="IBM437"/>
<xsd:enumeration value="IBM500"/>
<xsd:enumeration value="IBM775"/>
<xsd:enumeration value="IBM850"/>
<xsd:enumeration value="IBM851"/>
<xsd:enumeration value="IBM852"/>
<xsd:enumeration value="IBM855"/>
<xsd:enumeration value="IBM857"/>
<xsd:enumeration value="IBM860"/>
<xsd:enumeration value="IBM861"/>
```

```
<xsd:enumeration value="IBM862" />
<xsd:enumeration value="IBM863" />
<xsd:enumeration value="IBM864" />
<xsd:enumeration value="IBM865" />
<xsd:enumeration value="IBM866" />
<xsd:enumeration value="IBM868" />
<xsd:enumeration value="IBM869" />
<xsd:enumeration value="IBM870" />
<xsd:enumeration value="IBM871" />
<xsd:enumeration value="IBM880" />
<xsd:enumeration value="IBM891" />
<xsd:enumeration value="IBM903" />
<xsd:enumeration value="IBM904" />
<xsd:enumeration value="IBM905" />
<xsd:enumeration value="IBM918" />
<xsd:enumeration value="IBM1026" />
<xsd:enumeration value="EBCDIC-AT-DE" />
<xsd:enumeration value="EBCDIC-AT-DE-A" />
<xsd:enumeration value="EBCDIC-CA-FR" />
<xsd:enumeration value="EBCDIC-DK-NO" />
<xsd:enumeration value="EBCDIC-DK-NO-A" />
<xsd:enumeration value="EBCDIC-FI-SE" />
<xsd:enumeration value="EBCDIC-FI-SE-A" />
<xsd:enumeration value="EBCDIC-FR" />
<xsd:enumeration value="EBCDIC-IT" />
<xsd:enumeration value="EBCDIC-PT" />
<xsd:enumeration value="EBCDIC-ES" />
<xsd:enumeration value="EBCDIC-ES-A" />
<xsd:enumeration value="EBCDIC-ES-S" />
<xsd:enumeration value="EBCDIC-UK" />
<xsd:enumeration value="EBCDIC-US" />
<xsd:enumeration value="UNKNOWN-8BIT" />
<xsd:enumeration value="MNEMONIC" />
<xsd:enumeration value="MNEM" />
<xsd:enumeration value="VISCII" />
<xsd:enumeration value="VIQR" />
<xsd:enumeration value="KOI8-R" />
```

```
<xsd:enumeration value="KOI8-U" />
<xsd:enumeration value="IBM00858" />
<xsd:enumeration value="IBM00924" />
<xsd:enumeration value="IBM01140" />
<xsd:enumeration value="IBM01141" />
<xsd:enumeration value="IBM01142" />
<xsd:enumeration value="IBM01143" />
<xsd:enumeration value="IBM01144" />
<xsd:enumeration value="IBM01145" />
<xsd:enumeration value="IBM01146" />
<xsd:enumeration value="IBM01147" />
<xsd:enumeration value="IBM01148" />
<xsd:enumeration value="IBM01149" />
<xsd:enumeration value="Big5-HKSCS" />
<xsd:enumeration value="IBM1047" />
<xsd:enumeration value="PTCP154" />
<xsd:enumeration value="UNICODE-1-1" />
<xsd:enumeration value="SCSU" />
<xsd:enumeration value="UTF-7" />
<xsd:enumeration value="UTF-16BE" />
<xsd:enumeration value="UTF-16LE" />
<xsd:enumeration value="UTF-16" />
<xsd:enumeration value="CESU-8" />
<xsd:enumeration value="UTF-32" />
<xsd:enumeration value="UTF-32BE" />
<xsd:enumeration value="UTF-32LE" />
<xsd:enumeration value="BOCU-1" />
<xsd:enumeration value="UNICODE-1-1-UTF-7" />
<xsd:enumeration value="UTF-8" />
<xsd:enumeration value="ISO-8859-13" />
<xsd:enumeration value="ISO-8859-14" />
<xsd:enumeration value="ISO-8859-15" />
<xsd:enumeration value="ISO-8859-16" />
<xsd:enumeration value="GBK" />
<xsd:enumeration value="GB18030" />
<xsd:enumeration value="JIS_Encoding" />
<xsd:enumeration value="Shift_JIS" />
```

```
<xsd:enumeration value="Extended_UNIX_Code_Packed_Format_for_Japanese"/>
<xsd:enumeration value="Extended_UNIX_Code_Fixed_Width_for_Japanese"/>
<xsd:enumeration value="ISO-10646-UCS-Basic"/>
<xsd:enumeration value="ISO-10646-Unicode-Latin1"/>
<xsd:enumeration value="ISO-10646-J-1"/>
<xsd:enumeration value="ISO-Unicode-IBM-1261"/>
<xsd:enumeration value="ISO-Unicode-IBM-1268"/>
<xsd:enumeration value="ISO-Unicode-IBM-1276"/>
<xsd:enumeration value="ISO-Unicode-IBM-1264"/>
<xsd:enumeration value="ISO-Unicode-IBM-1265"/>
<xsd:enumeration value="ISO-8859-1-Windows-3.0-Latin-1"/>
<xsd:enumeration value="ISO-8859-1-Windows-3.1-Latin-1"/>
<xsd:enumeration value="ISO-8859-2-Windows-Latin-2"/>
<xsd:enumeration value="ISO-8859-9-Windows-Latin-5"/>
<xsd:enumeration value="Adobe-Standard-Encoding"/>
<xsd:enumeration value="Ventura-US"/>
<xsd:enumeration value="Ventura-International"/>
<xsd:enumeration value="PC8-Danish-Norwegian"/>
<xsd:enumeration value="PC8-Turkish"/>
<xsd:enumeration value="IBM-Symbols"/>
<xsd:enumeration value="IBM-Thai"/>
<xsd:enumeration value="HP-Legal"/>
<xsd:enumeration value="HP-Pi-font"/>
<xsd:enumeration value="HP-Math8"/>
<xsd:enumeration value="Adobe-Symbol-Encoding"/>
<xsd:enumeration value="HP-DeskTop"/>
<xsd:enumeration value="Ventura-Math"/>
<xsd:enumeration value="Microsoft-Publishing"/>
<xsd:enumeration value="Windows-31J"/>
<xsd:enumeration value="GB2312"/>
<xsd:enumeration value="Big5"/>
<xsd:enumeration value="windows-1250"/>
<xsd:enumeration value="windows-1251"/>
<xsd:enumeration value="windows-1252"/>
<xsd:enumeration value="windows-1253"/>
<xsd:enumeration value="windows-1254"/>
<xsd:enumeration value="windows-1255"/>
```



```
<xsd:enumeration value="windows-1256" />
<xsd:enumeration value="windows-1257" />
<xsd:enumeration value="windows-1258" />
<xsd:enumeration value="TIS-620" />
<xsd:enumeration value="HZ-GB-2312" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

## Appendix H. XML Schema for Media Object 3D Studio Max Project AMS

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

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:arco="http://www.arco-
web.org/namespaces/arco/v10" elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xsd:annotation>

    <xsd:documentation>This schema was developed by the ARCO consortium for the 2nd
    prototype of the ARCO project. Date 22-10-02. Developed by Nick Mourkoussis
    n.mourkoussis@sussex.ac.uk. People involved: Jacek Chmielewski
    jchmiel@kti.ae.poznan.pl, Krzysztof Walczak walczak@kti.ae.poznan.pl</xsd:documentation>

  </xsd:annotation>

  <xsd:element name="mediaObject3dsMaxProject">

    <xsd:annotation>

      <xsd:appinfo>

        <arco:displayName>3ds Max Project Media Object</arco:displayName>

        <arco:definition>AMS schema described by metadata associated with 3ds max
        project media objects</arco:definition>

      </xsd:appinfo>

    </xsd:annotation>

    <xsd:complexType>

      <xsd:sequence>

        <xsd:element name="technique" maxOccurs="unbounded">

          <xsd:annotation>

            <xsd:appinfo>

              <arco:displayName>Technique</arco:displayName>

              <arco:identifier>http://www.arco-
              web.org/schemas/arco/technique</arco:identifier>

              <arco:definition>The technique used to acquire a digital manifestation of
              the Cultural Object</arco:definition>

              <arco:content>This should be one item from predefined dictionary.Manual
              Modelling, Image based modelling</arco:content>

            </xsd:appinfo>

          </xsd:annotation>

          <xsd:simpleType>

            <xsd:restriction base="xsd:string">

              <xsd:enumeration value="Image Based Modelling"/>

              <xsd:enumeration value="Manual Modelling"/>

            </xsd:restriction>

          </xsd:simpleType>

        </xsd:element>

      </xsd:sequence>

    </xsd:complexType>

  </xsd:element>

</xsd:schema>
```

```
<xsd:element name="softwareVersion" type="allVersions">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:displayName>Software version</arco:displayName>
      <arco:identifier>http://www.arco-web.org/schemas/arco/software-
version</arco:identifier>
      <arco:definition>Version of 3D Studio MAX software used to save this
file</arco:definition>
      <arco:content>This should be one from predefined dictionary.3.0, 4.0, 4.2,
5.0</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>

<xsd:element name="requiredExtensions" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:displayName>Required Extensions</arco:displayName>
      <arco:identifier>http://www.arco-web.org/schemas/arco/required-
extensions</arco:identifier>
      <arco:definition>Extensions and plug-ins needed to properly open and
display this file</arco:definition>
      <arco:content>Names of needed extensions separated by comma</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<xsd:simpleType name="existingVersions">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="3.0"/>
    <xsd:enumeration value="4.0"/>
    <xsd:enumeration value="4.2"/>
    <xsd:enumeration value="5.0"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="anyPossibleVersions">
  <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:simpleType name="allVersions">
```

```
<xsd:union memberTypes="existingVersions anyPossibleVersions" />  
</xsd:simpleType>  
</xsd:schema>
```

## Appendix I. XML Schema for Media Object VRML Model AMS

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

<xsd:schema xmlns:arco="http://www.arco-web.org/namespaces/arco/v10"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xsd:annotation>

    <xsd:documentation>This schema was developed by the ARCO consortium for the 2nd
prototype of the ARCO project. Date 22-10-02. Developed by Nick Mourkoussis
n.mourkoussis@sussex.ac.uk. People involved: Jacek Chmielewski
jchmiel@kti.ae.poznan.pl, Krzysztof Walczak walczak@kti.ae.poznan.pl</xsd:documentation>

  </xsd:annotation>

  <xsd:element name="mediaObjectVRMLModel">

    <xsd:annotation>

      <xsd:appinfo>

        <arco:displayName>VRML Model Media Object</arco:displayName>

        <arco:definition>AMS schema described by metadata associated with VRML model
media objects</arco:definition>

      </xsd:appinfo>

    </xsd:annotation>

    <xsd:complexType>

      <xsd:sequence>

        <xsd:element name="technique" maxOccurs="unbounded">

          <xsd:annotation>

            <xsd:appinfo>

              <arco:displayName>Technique</arco:displayName>

              <arco:identifier>http://www.arco-
web.org/schemas/arco/technique</arco:identifier>

              <arco:definition>The technique used to acquire a digital manifestation of
the Cultural Object</arco:definition>

              <arco:content>This should be one item from predefined dictionary.Image
based Modelling, Manual Modelling, 3ds max export</arco:content>

            </xsd:appinfo>

          </xsd:annotation>

          <xsd:simpleType>

            <xsd:restriction base="xsd:string">

              <xsd:enumeration value="3DMAX export"/>

              <xsd:enumeration value="Image based Modelling"/>

              <xsd:enumeration value="Manual Modelling"/>

            </xsd:restriction>

          </xsd:simpleType>

        </xsd:element>

      </xsd:sequence>

    </xsd:complexType>

  </xsd:element>

</xsd:schema>
```

```
</xsd:element>

<xsd:element name="vrmVersion">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>MOT_VRMLModel_VRMLVersion</arco:autoGenerated>
      <arco:displayName>VRML Version</arco:displayName>
      <arco:identifier>http://www.arco-web.org/schemas/arco/vrml-
version</arco:identifier>
      <arco:definition>Version of VRML language used by the
model</arco:definition>
      <arco:content>Not Editable</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="1.0"/>
      <xsd:enumeration value="2.0"/>
      <xsd:enumeration value="97"/>
      <xsd:enumeration value="200x"/>
      <xsd:enumeration value="x3D"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="numberOfTextures" type="xsd:integer">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>MOT_VRMLModel_NumberOfTextures</arco:autoGenerated>
      <arco:displayName>Number of Textures</arco:displayName>
      <arco:identifier>http://www.arco-
web.org/schemas/arco/textures</arco:identifier>
      <arco:definition>Number of textures used by this VRML
model</arco:definition>
      <arco:content>Not Editable</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>

<xsd:element name="composite" type="xsd:boolean">
  <xsd:annotation>
    <xsd:appinfo>
```

```
<arco:autoGenerated>MOT_VRMLModel_Composite</arco:autoGenerated>

<arco:displayName>Composite</arco:displayName>

<arco:identifier>http://www.arco-
web.org/schemas/arco/composite</arco:identifier>

<arco:definition>Indicates whether the model is composed of more than one
VRML file</arco:definition>

<arco:content>Not Editable</arco:content>

</xsd:appinfo>

</xsd:annotation>

</xsd:element>

<xsd:element name="animated" type="xsd:boolean">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>MOT_VRMLModel_Animated</arco:autoGenerated>
      <arco:displayName>Animated</arco:displayName>
      <arco:identifier>http://www.arco-
web.org/schemas/arco/animated</arco:identifier>
      <arco:definition>Indicates whether the model contains animated
elements</arco:definition>
      <arco:content>Not Editable</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

## Appendix J. XML Schema for Media Object Panorama Image AMS

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

<xsd:schema xmlns:arco="http://www.arco-web.org/namespaces/arco/v10"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
  attributeFormDefault="unqualified">

  <xsd:annotation>

    <xsd:documentation>This schema was developed by the ARCO consortium for the 2nd
      prototype of the ARCO project. Date 22-10-02. Developed by Nick Mourkoussis
      n.mourkoussis@sussex.ac.uk. People involved: Jacek Chmielewski
      jchmiel@kti.ae.poznan.pl, Krzysztof Walczak walczak@kti.ae.poznan.pl</xsd:documentation>

    </xsd:annotation>

    <xsd:element name="mediaObjectPanoramaImage">

      <xsd:annotation>

        <xsd:appinfo>

          <arco:displayName>Panaroma Image Media Object</arco:displayName>

          <arco:definition>AMS schema described by metadata associated with panorama
            images media objects</arco:definition>

        </xsd:appinfo>

      </xsd:annotation>

      <xsd:complexType>

        <xsd:sequence>

          <xsd:element name="technique" maxOccurs="unbounded">

            <xsd:annotation>

              <xsd:appinfo>

                <arco:displayName>Technique</arco:displayName>

                <arco:identifier>http://www.arco-
                  web.org/schemas/arco/technique</arco:identifier>

                <arco:definition>The technique used to acquire a panorama image of the
                  Cultural Object</arco:definition>

                <arco:content>This should be one item from predefined dictionary.Human
                  image processing, Automated image processing</arco:content>

              </xsd:appinfo>

            </xsd:annotation>

            <xsd:simpleType>

              <xsd:restriction base="xsd:string">

                <xsd:enumeration value="Automated Image Processing"/>

                <xsd:enumeration value="Human Image Processing"/>

              </xsd:restriction>

            </xsd:simpleType>

          </xsd:element>

        </xsd:sequence>

      </xsd:complexType>

    </xsd:element>

  </xsd:annotation>

</xsd:schema>
```



```
<xsd:element name="numberOfImages" type="xsd:integer">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>MOT_PanoramaImage_NumberOfImages</arco:autoGenerated>
      <arco:displayName>Number of images</arco:displayName>
      <arco:identifier>http://www.arco-web.org/schemas/arco/number-
images</arco:identifier>
      <arco:definition>Number of images included in this panorama
view</arco:definition>
      <arco:content>Not Editable</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>

<xsd:element name="stepAngle" type="xsd:float">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:autoGenerated>MOT_PanoramaImage_StepAngle</arco:autoGenerated>
      <arco:displayName>Step angle</arco:displayName>
      <arco:identifier>http://www.arco-web.org/schemas/arco/step-
angle</arco:identifier>
      <arco:definition>Step angle between images</arco:definition>
      <arco:content>Positive and negative values can be used. It is assumed that
the rotation axis points in the image-up direction</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

## Appendix K. XML Schema for Media Object Multiresolution Image AMS

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

<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:arco="http://www.arco-
web.org/namespaces/arco/v10" elementFormDefault="qualified"
attributeFormDefault="unqualified">

  <xsd:annotation>

    <xsd:documentation>This schema was developed by the ARCO consortium for the 2nd
    prototype of the ARCO project. Date 22-10-02. Developed by Nick Mourkoussis
    n.mourkoussis@sussex.ac.uk. People involved: Jacek Chmielewski
    jchmiel@kti.ae.poznan.pl, Krzysztof Walczak walczak@kti.ae.poznan.pl</xsd:documentation>

  </xsd:annotation>

  <xsd:element name="mediaObjectMultiresolutionImage">

    <xsd:annotation>

      <xsd:appinfo>

        <arco:displayName>Multiresolution Image Media Object</arco:displayName>

        <arco:definition>AMS schema described by metadata associated with
        Multiresolution Image media objects</arco:definition>

      </xsd:appinfo>

    </xsd:annotation>

    <xsd:complexType>

      <xsd:sequence>

        <xsd:element name="technique" maxOccurs="unbounded">

          <xsd:annotation>

            <xsd:appinfo>

              <arco:displayName>Technique</arco:displayName>

              <arco:identifier>http://www.arco-
              web.org/schemas/arco/multiresolution</arco:identifier>

              <arco:definition>The technique used to acquire an multiresolution image of
              the Cultural Object</arco:definition>

              <arco:content>This should be one item from predefined dictionary.Human
              image processing, Automated image processing. ACMA wizard</arco:content>

            </xsd:appinfo>

          </xsd:annotation>

          <xsd:simpleType>

            <xsd:restriction base="xsd:string">

              <xsd:enumeration value="ACMA wizard"/>

              <xsd:enumeration value="Automated image processing"/>

              <xsd:enumeration value="Human image processing"/>

            </xsd:restriction>

          </xsd:simpleType>

        </xsd:sequence>

      </xsd:complexType>

    </xsd:element>

  </xsd:element>

</xsd:schema>
```

```
</xsd:element>

<xsd:element name="resolutions">
  <xsd:annotation>
    <xsd:appinfo>

<arco:autoGenerated>MOT_MultiresolutionImage_Resolutions</arco:autoGenerated>

    <arco:displayName>Resolutions</arco:displayName>
    <arco:identifier>http://www.arco-
web.org/schemas/arco/resolutions</arco:identifier>
    <arco:definition>Available image sizes (width x height in
pixels)</arco:definition>
    <arco:content>Not Editable</arco:content>
  </xsd:appinfo>
</xsd:annotation>
<xsd:simpleType>
  <xsd:list>
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="\d+x\d+"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:list>
</xsd:simpleType>
</xsd:element>

<xsd:element name="software" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:displayName>Software</arco:displayName>
      <arco:identifier>http://www.arco-
web.org/schemas/arco/software</arco:identifier>
      <arco:definition>Software used for generating the MR
image</arco:definition>
      <arco:content>Application name</arco:content>
    </xsd:appinfo>
  </xsd:annotation>
</xsd:element>

<xsd:element name="algorithm" type="xsd:string" minOccurs="0">
  <xsd:annotation>
    <xsd:appinfo>
      <arco:displayName>Algorithm</arco:displayName>
```

```
<arco:identifier>http://www.arco-  
web.org/schemas/arco/algorithm</arco:identifier>  
  
<arco:definition>Algorithm used for rescaling original  
image</arco:definition>  
  
<arco:content>Algorithm name</arco:content>  
  
</xsd:appinfo>  
  
</xsd:annotation>  
  
</xsd:element>  
  
</xsd:sequence>  
  
</xsd:complexType>  
  
</xsd:element>  
  
</xsd:schema>
```

## Appendix L. XML Schema for characterSet element

```
<xsd:element name="characterSet" minOccurs="0">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="ANSI_X3.4-1968"/>
      <xsd:enumeration value="ISO-10646-UTF-1"/>
      <xsd:enumeration value="INVARIANT"/>
      <xsd:enumeration value="ISO_646.irv:1983"/>
      <xsd:enumeration value="BS_4730"/>
      <xsd:enumeration value="NATS-SEFI"/>
      <xsd:enumeration value="NATS-SEFI-ADD"/>
      <xsd:enumeration value="NATS-DANO"/>
      <xsd:enumeration value="NATS-DANO-ADD"/>
      <xsd:enumeration value="SEN_850200_B"/>
      <xsd:enumeration value="SEN_850200_C"/>
      <xsd:enumeration value="KS_C_5601-1987"/>
      <xsd:enumeration value="ISO-2022-KR"/>
      <xsd:enumeration value="EUC-KR"/>
      <xsd:enumeration value="ISO-2022-JP"/>
      <xsd:enumeration value="ISO-2022-JP-2"/>
      <xsd:enumeration value="ISO-2022-CN"/>
      <xsd:enumeration value="ISO-2022-CN-EXT"/>
      <xsd:enumeration value="JIS_C6220-1969-jp"/>
      <xsd:enumeration value="JIS_C6220-1969-ro"/>
      <xsd:enumeration value="IT"/>
      <xsd:enumeration value="PT"/>
      <xsd:enumeration value="ES"/>
      <xsd:enumeration value="greek7-old"/>
      <xsd:enumeration value="latin-greek"/>
      <xsd:enumeration value="DIN_66003"/>
      <xsd:enumeration value="NF_Z_62-010_(1973)"/>
      <xsd:enumeration value="Latin-greek-1"/>
      <xsd:enumeration value="ISO_5427"/>
      <xsd:enumeration value="JIS_C6226-1978"/>
      <xsd:enumeration value="BS_viewdata"/>
      <xsd:enumeration value="INIS"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

```
<xsd:enumeration value="INIS-8"/>
<xsd:enumeration value="INIS-cyrillic"/>
<xsd:enumeration value="ISO_5427:1981"/>
<xsd:enumeration value="ISO_5428:1980"/>
<xsd:enumeration value="GB_1988-80"/>
<xsd:enumeration value="GB_2312-80"/>
<xsd:enumeration value="NS_4551-1"/>
<xsd:enumeration value="NS_4551-2"/>
<xsd:enumeration value="NF_Z_62-010"/>
<xsd:enumeration value="videotex-suppl"/>
<xsd:enumeration value="PT2"/>
<xsd:enumeration value="ES2"/>
<xsd:enumeration value="MSZ_7795.3"/>
<xsd:enumeration value="JIS_C6226-1983"/>
<xsd:enumeration value="greek7"/>
<xsd:enumeration value="ASMO_449"/>
<xsd:enumeration value="iso-ir-90"/>
<xsd:enumeration value="JIS_C6229-1984-a"/>
<xsd:enumeration value="JIS_C6229-1984-b"/>
<xsd:enumeration value="JIS_C6229-1984-b-add"/>
<xsd:enumeration value="JIS_C6229-1984-hand"/>
<xsd:enumeration value="JIS_C6229-1984-hand-add"/>
<xsd:enumeration value="JIS_C6229-1984-kana"/>
<xsd:enumeration value="ISO_2033-1983"/>
<xsd:enumeration value="ANSI_X3.110-1983"/>
<xsd:enumeration value="ISO_8859-1:1987"/>
<xsd:enumeration value="ISO_8859-2:1987"/>
<xsd:enumeration value="T.61-7bit"/>
<xsd:enumeration value="T.61-8bit"/>
<xsd:enumeration value="ISO_8859-3:1988"/>
<xsd:enumeration value="ISO_8859-4:1988"/>
<xsd:enumeration value="ECMA-cyrillic"/>
<xsd:enumeration value="CSA_Z243.4-1985-1"/>
<xsd:enumeration value="CSA_Z243.4-1985-2"/>
<xsd:enumeration value="CSA_Z243.4-1985-gr"/>
<xsd:enumeration value="ISO_8859-6:1987"/>
<xsd:enumeration value="ISO_8859-6-E"/>
```

```
<xsd:enumeration value="ISO_8859-6-I" />
<xsd:enumeration value="ISO_8859-7:1987" />
<xsd:enumeration value="T.101-G2" />
<xsd:enumeration value="ISO_8859-8:1988" />
<xsd:enumeration value="ISO_8859-8-E" />
<xsd:enumeration value="ISO_8859-8-I" />
<xsd:enumeration value="CSN_369103" />
<xsd:enumeration value="JUS_I.B1.002" />
<xsd:enumeration value="ISO_6937-2-add" />
<xsd:enumeration value="IEC_P27-1" />
<xsd:enumeration value="ISO_8859-5:1988" />
<xsd:enumeration value="JUS_I.B1.003-serb" />
<xsd:enumeration value="JUS_I.B1.003-mac" />
<xsd:enumeration value="ISO_8859-9:1989" />
<xsd:enumeration value="greek-ccitt" />
<xsd:enumeration value="NC_NC00-10:81" />
<xsd:enumeration value="ISO_6937-2-25" />
<xsd:enumeration value="GOST_19768-74" />
<xsd:enumeration value="ISO_8859-supp" />
<xsd:enumeration value="ISO_10367-box" />
<xsd:enumeration value="ISO-8859-10" />
<xsd:enumeration value="latin-lap" />
<xsd:enumeration value="JIS_X0212-1990" />
<xsd:enumeration value="DS_2089" />
<xsd:enumeration value="us-dk" />
<xsd:enumeration value="dk-us" />
<xsd:enumeration value="JIS_X0201" />
<xsd:enumeration value="ISO-10646-UCS-2" />
<xsd:enumeration value="DEC-MCS" />
<xsd:enumeration value="hp-roman8" />
<xsd:enumeration value="macintosh" />
<xsd:enumeration value="IBM037" />
<xsd:enumeration value="IBM038" />
<xsd:enumeration value="IBM273" />
<xsd:enumeration value="IBM274" />
<xsd:enumeration value="IBM275" />
<xsd:enumeration value="IBM278" />
```

```
<xsd:enumeration value="IBM280" />
<xsd:enumeration value="IBM281" />
<xsd:enumeration value="IBM284" />
<xsd:enumeration value="IBM285" />
<xsd:enumeration value="IBM290" />
<xsd:enumeration value="IBM297" />
<xsd:enumeration value="IBM420" />
<xsd:enumeration value="IBM423" />
<xsd:enumeration value="IBM424" />
<xsd:enumeration value="IBM437" />
<xsd:enumeration value="IBM500" />
<xsd:enumeration value="IBM775" />
<xsd:enumeration value="IBM850" />
<xsd:enumeration value="IBM851" />
<xsd:enumeration value="IBM852" />
<xsd:enumeration value="IBM855" />
<xsd:enumeration value="IBM857" />
<xsd:enumeration value="IBM860" />
<xsd:enumeration value="IBM861" />
<xsd:enumeration value="IBM862" />
<xsd:enumeration value="IBM863" />
<xsd:enumeration value="IBM864" />
<xsd:enumeration value="IBM865" />
<xsd:enumeration value="IBM866" />
<xsd:enumeration value="IBM868" />
<xsd:enumeration value="IBM869" />
<xsd:enumeration value="IBM870" />
<xsd:enumeration value="IBM871" />
<xsd:enumeration value="IBM880" />
<xsd:enumeration value="IBM891" />
<xsd:enumeration value="IBM903" />
<xsd:enumeration value="IBM904" />
<xsd:enumeration value="IBM905" />
<xsd:enumeration value="IBM918" />
<xsd:enumeration value="IBM1026" />
<xsd:enumeration value="EBCDIC-AT-DE" />
<xsd:enumeration value="EBCDIC-AT-DE-A" />
```



```
<xsd:enumeration value="EBCDIC-CA-FR" />
<xsd:enumeration value="EBCDIC-DK-NO" />
<xsd:enumeration value="EBCDIC-DK-NO-A" />
<xsd:enumeration value="EBCDIC-FI-SE" />
<xsd:enumeration value="EBCDIC-FI-SE-A" />
<xsd:enumeration value="EBCDIC-FR" />
<xsd:enumeration value="EBCDIC-IT" />
<xsd:enumeration value="EBCDIC-PT" />
<xsd:enumeration value="EBCDIC-ES" />
<xsd:enumeration value="EBCDIC-ES-A" />
<xsd:enumeration value="EBCDIC-ES-S" />
<xsd:enumeration value="EBCDIC-UK" />
<xsd:enumeration value="EBCDIC-US" />
<xsd:enumeration value="UNKNOWN-8BIT" />
<xsd:enumeration value="MNEMONIC" />
<xsd:enumeration value="MNEM" />
<xsd:enumeration value="VISCII" />
<xsd:enumeration value="VIQR" />
<xsd:enumeration value="KOI8-R" />
<xsd:enumeration value="KOI8-U" />
<xsd:enumeration value="IBM00858" />
<xsd:enumeration value="IBM00924" />
<xsd:enumeration value="IBM01140" />
<xsd:enumeration value="IBM01141" />
<xsd:enumeration value="IBM01142" />
<xsd:enumeration value="IBM01143" />
<xsd:enumeration value="IBM01144" />
<xsd:enumeration value="IBM01145" />
<xsd:enumeration value="IBM01146" />
<xsd:enumeration value="IBM01147" />
<xsd:enumeration value="IBM01148" />
<xsd:enumeration value="IBM01149" />
<xsd:enumeration value="Big5-HKSCS" />
<xsd:enumeration value="IBM1047" />
<xsd:enumeration value="PTCP154" />
<xsd:enumeration value="UNICODE-1-1" />
<xsd:enumeration value="SCSU" />
```

```
<xsd:enumeration value="UTF-7"/>
<xsd:enumeration value="UTF-16BE"/>
<xsd:enumeration value="UTF-16LE"/>
<xsd:enumeration value="UTF-16"/>
<xsd:enumeration value="CESU-8"/>
<xsd:enumeration value="UTF-32"/>
<xsd:enumeration value="UTF-32BE"/>
<xsd:enumeration value="UTF-32LE"/>
<xsd:enumeration value="BOCU-1"/>
<xsd:enumeration value="UNICODE-1-1-UTF-7"/>
<xsd:enumeration value="UTF-8"/>
<xsd:enumeration value="ISO-8859-13"/>
<xsd:enumeration value="ISO-8859-14"/>
<xsd:enumeration value="ISO-8859-15"/>
<xsd:enumeration value="ISO-8859-16"/>
<xsd:enumeration value="GBK"/>
<xsd:enumeration value="GB18030"/>
<xsd:enumeration value="JIS_Encoding"/>
<xsd:enumeration value="Shift_JIS"/>
<xsd:enumeration value="Extended_UNIX_Code_Packed_Format_for_Japanese"/>
<xsd:enumeration value="Extended_UNIX_Code_Fixed_Width_for_Japanese"/>
<xsd:enumeration value="ISO-10646-UCS-Basic"/>
<xsd:enumeration value="ISO-10646-Unicode-Latin1"/>
<xsd:enumeration value="ISO-10646-J-1"/>
<xsd:enumeration value="ISO-Unicode-IBM-1261"/>
<xsd:enumeration value="ISO-Unicode-IBM-1268"/>
<xsd:enumeration value="ISO-Unicode-IBM-1276"/>
<xsd:enumeration value="ISO-Unicode-IBM-1264"/>
<xsd:enumeration value="ISO-Unicode-IBM-1265"/>
<xsd:enumeration value="ISO-8859-1-Windows-3.0-Latin-1"/>
<xsd:enumeration value="ISO-8859-1-Windows-3.1-Latin-1"/>
<xsd:enumeration value="ISO-8859-2-Windows-Latin-2"/>
<xsd:enumeration value="ISO-8859-9-Windows-Latin-5"/>
<xsd:enumeration value="Adobe-Standard-Encoding"/>
<xsd:enumeration value="Ventura-US"/>
<xsd:enumeration value="Ventura-International"/>
<xsd:enumeration value="PC8-Danish-Norwegian"/>
```

```
<xsd:enumeration value="PC8-Turkish" />
<xsd:enumeration value="IBM-Symbols" />
<xsd:enumeration value="IBM-Thai" />
<xsd:enumeration value="HP-Legal" />
<xsd:enumeration value="HP-Pi-font" />
<xsd:enumeration value="HP-Math8" />
<xsd:enumeration value="Adobe-Symbol-Encoding" />
<xsd:enumeration value="HP-DeskTop" />
<xsd:enumeration value="Ventura-Math" />
<xsd:enumeration value="Microsoft-Publishing" />
<xsd:enumeration value="Windows-31J" />
<xsd:enumeration value="GB2312" />
<xsd:enumeration value="Big5" />
<xsd:enumeration value="windows-1250" />
<xsd:enumeration value="windows-1251" />
<xsd:enumeration value="windows-1252" />
<xsd:enumeration value="windows-1253" />
<xsd:enumeration value="windows-1254" />
<xsd:enumeration value="windows-1255" />
<xsd:enumeration value="windows-1256" />
<xsd:enumeration value="windows-1257" />
<xsd:enumeration value="windows-1258" />
<xsd:enumeration value="TIS-620" />
<xsd:enumeration value="HZ-GB-2312" />
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
```

## Appendix M. Sample XSL Stylesheet for AMS Metadata

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

<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xsl:template match="/CulturalObjectAMS">

    <html>

      <head/>

      <body>

        <span style="background-color:black; color:aqua; font-family:Verdana; font-
size:14pt; font-style:normal; font-weight:normal">AMS data for : </span>

        <xsl:for-each select="CulturalObject">

          <xsl:for-each select="Name">

            <span style="font-family:Verdana; font-weight:bold">

              "<xsl:apply-templates/>"

            </span>

          </xsl:for-each>

        </xsl:for-each>

        <br/>

        <br/>

        <img border="0" style="background-color:black; color:aqua">

          <xsl:attribute name="src"><xsl:text disable-output-
escaping="yes">http://arco.kti.ae.poznan.pl/arif/adam/arco/getto?id=103</xsl:text></xsl:
attribute>

        </img>

        <span style="background-color:black; color:aqua; font-family:Verdana; font-
size:12pt; font-weight:normal"> Creator : </span>

        <xsl:for-each select="CulturalObject">

          <xsl:for-each select="Creator">

            <span style="color:#FF8000; font-family:Verdana; font-weight:bold">

              <xsl:apply-templates/>

            </span>

          </xsl:for-each>

        </xsl:for-each>

        <br/>

        <img border="0" style="background-color:black; color:aqua; font-family:Verdana;
font-size:smaller; font-weight:normal">

          <xsl:attribute name="src"><xsl:text disable-output-
escaping="yes">http://arco.kti.ae.poznan.pl/arif/adam/arco/getto?id=103</xsl:text></xsl:
attribute>
```

```
</img>

<span style="background-color:black; color:aqua; font-family:Verdana; font-
size:12pt; font-weight:normal"> Creation date : </span>

<xsl:for-each select="CulturalObject">

  <xsl:for-each select="DateCreated">

    <span style="color:#FF8000; font-family:Verdana; font-weight:bold">

      <xsl:apply-templates/>

    </span>

  </xsl:for-each>

</xsl:for-each>

<br/>

<br/>

<img border="0" style="background-color:black; color:aqua; font-family:Verdana;
font-size:smaller; font-weight:normal">

  <xsl:attribute name="src"><xsl:text disable-output-
escaping="yes">http://arco.kti.ae.poznan.pl/arif/adam/arco/getto?id=103</xsl:text></xsl:
attribute>

</img>

<span style="background-color:black; color:aqua; font-family:Verdana; font-
size:12pt; font-weight:normal"> Description : </span>

<br/>

<xsl:for-each select="CulturalObject">

  <xsl:for-each select="Description">

    <span style="color:#FF8000; font-family:Verdana; font-size:11pt">

      <xsl:apply-templates/>

    </span>

  </xsl:for-each>

</xsl:for-each>

<br/>

<br/>

</body>

</html>

</xsl:template>

</xsl:stylesheet>
```