IST IST-2000-28336 **ARCO**

Augmented Representation of Cultural Objects

XML Metadata, Schemas and XSL

Final Report on XML Technology

Author: PUE, UoS, UKOLN

Issued by: ARCO Consortium

Version: 1.0, Release

Date: 29-Sep-2003

Report Documentation Page

Report Documentation

Period	4
Distribution	Paper, Web Site, and Email
Work Package	WP6
Deliverable	D11
Security	Public
Project Number	IST-2000-28336
File Name	ARCO-D11-R-1.0-290903.doc

Report Change Log

Version	Author(s)	Description	Date
0.1, Draft	K. Walczak (PUE)	Initial draft	12-Aug-2003
0.2, Draft	K. Walczak, O. Huminiecki (PUE)	Updated XDE implementation	13-Aug-2003
0.3, Draft	K. Walczak (PUE)	Integration	22-Aug-2003
0.4, Draft	J. Chmielewski (PUE)	Updated AMS implementation	25-Aug-2003
0.5, Draft	M. Stawniak (PUE)	Updated Template Manager and Presentation Manager	26-Aug-2003
0.6, Draft	K. Walczak (PUE)	Updated X-VRML	26-Aug-2003
0.7, Draft	W. Wiza (PUE)	Updated XSL description	27-Aug-2003
0.8, Draft	K. Walczak (PUE)	Final edits	27-Aug-2003
0.9, Draft	M. Patel (UKOLN)	Modifications to application profiles section + took out AMS element Owner as per AMS Specification	01-Sept-2003
0.10, Draft	N. Mourkoussis (UoS)	Modifications to section 2 according to the final AMS specification and implementation. Updated Appendices A to K	04-Sep-2003
0.11, Final Draft	J. Chmielewski (PUE)	Updated styles, layout and references	08-Sep-2003
1.0, Release	M. White (UoS)	Final check and release	29-Sep-2003

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

Table of Contents

List of Figures and Tables

Summary

This document describes a suite of XML related technologies and tools developed and used within the ARCO project. These include the XML Schemas for AMS metadata descriptions of cultural objects, the X-VRML language used to dynamically build advanced augmented reality end-user interfaces (ARIF interfaces), the XSL templates used to format the AMS metadata for display in the ARIF interfaces, and the XDE – XML Data Exchange format being a common data interface between the ARCO components and between the ARCO system and external systems and applications.

This public report describes the status of these technologies developed for the final ARCO prototype, i.e. after the first two years of the project. In the final ARCO system the particular implementation elements may be different; however, no major architectural or design changes are expected.

1. Introduction

This document describes a suite of XML related technologies and tools developed and used within the ARCO project. These include the XML Schemas for AMS metadata descriptions of cultural objects, the X-VRML language used to dynamically build advanced augmented reality end-user interfaces (ARIF interfaces), the XSL templates used to format the AMS metadata for display in the ARIF interfaces, and the XDE – XML Data Exchange format being a common data interface between the ARCO components and between the ARCO system and external systems and applications.

This public report describes the status of these technologies developed for the final ARCO prototype, i.e. after the first two years of the project. In the final ARCO system the particular implementation elements may be different; however, no major architectural or design changes are expected.

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 "Report on the XML descriptions of the database cultural objects" [1]. 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 described. An overview of the use of XSL stylesheets in ARCO is also provided. The XSL stylesheets are used for displaying the AMS XML data in the Web ARIF interfaces.

The third part of the document describes the XDE – XML Data Exchange format. XDE is a common XML-based interface format used by all ARCO system components. Use of XDE enables close integration of all ARCO tools into a coherent suite, at the same time providing communication mechanisms that make the ARCO system inherently extensible and open for communication with other systems and applications.

2. XML Schema for AMS

2.1. Introduction

In this section, the XML implementation of the ARCO AMS metadata element sets defined in ARCO AMS Specification – "Report on the XML descriptions of the database of 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.

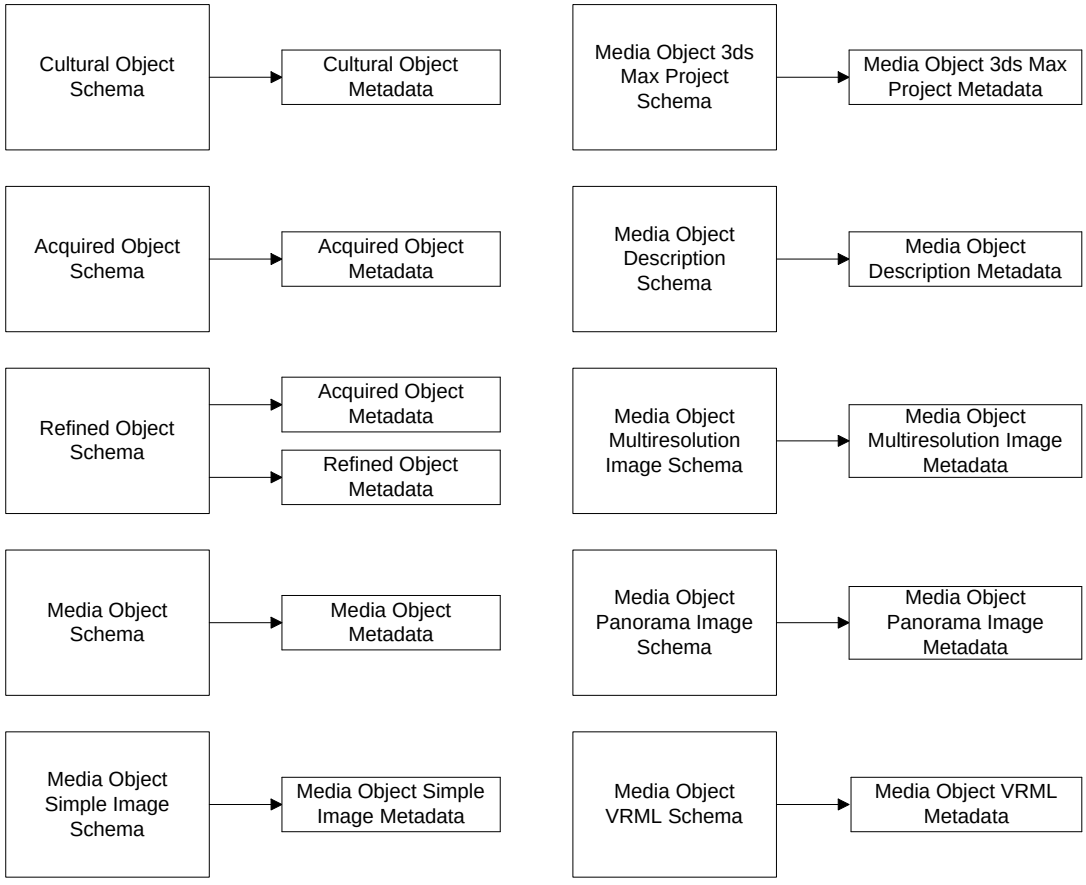


Figure 1. The structure of AMS Schema

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], which 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. The appropriate XML Schema implementation is provided in Appendix A.

2.2.2. AMS Schemas Namespaces

”A schema can be viewed as a collection of type definitions and element declarations whose names belong to a particular namespace called a target namespace” [21]. In ARCO there are several cases where the same element is used in different schemas. For example the element description is used in all CO, AO, RO, MO schemas. Even it looks like it is the same element in reality it is not. In the case it used in the CO schema it is used to define a description of the actual artefact. When it used within the Acquired object schema it is used to define a description for the acquired object. That is way in ARCO identified several distinct namespaces based on the ARCO web space in order to register our schemas. Those namespaces and their prefixes are presented in the table below.

Schema Name	**Target Namespace**	**Prefix**
Arco Application Elements	http://www.arco-web.org/schemas/vresion11/application/	arco
Administrative Metadata	http://www.arco-web.org/schemas/vresion11/adminstrative/	adm
Cultural Object	http://www.arco-web.org/schemas/vresion11/cultural/	clt
Acquired Object	http://www.arco-web.org/schemas/vresion11/acquired/	acq
Refined Object	http://www.arco-web.org/schemas/vresion11/refined/	rfn
Media Object	http://www.arco-web.org/schemas/vresion11/media/	mdo
Media Object Type Simple Image	http://www.arco-web.org/schemas/vresion11/image/	msi
Media Object Type 3ds MAX Project	http://www.arco-web.org/schemas/vresion11/max/	mde
Media Object Type Description	http://www.arco-web.org/schemas/vresion11/description/	mmx
Media Object Type Multiresolution Image	http://www.arco-web.org/schemas/vresion11/vrml/	vrm

Media Object Type Panoramic Image	http://www.arco-web.org/schemas/vresion11/panoramicc/	mpi
Media Object Type VRML Model	http://www.arco-web.org/schemas/vresion11/multiresolution/	mmi

Table 1. Namespaces of the ARCO schemas

2.2.3. Naming Convention

Each 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 [17]:

```
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.

2.2.4. 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 indicates 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).

Example

Another element included in the <appinfo> section of the element annotations is *example*. The example can be used by the end-user interface in order to provide possible examples of contextual values for each element (see Figure 3).

Use Type

Another element included in the <appinfo> section of the element annotations is *use type*. The use type can be used by the end-user interface in order to provide possible uses of information resources (see Figure 3). Within ARCO we have defined two possible uses of imformation resources. [1]

```
<xsd:element name="title" 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>This may be a common name or classification of an object in a textual or
codified form.</arco:content>
         <arco:useType>Intelligence</arco:useType>
         <arco:example>e.g. Castor || Mortar || Tray</arco:example>
      </xsd:appinfo>
   </xsd:annotation>
</xsd:element>
```

Figure 3. Encoding of additional information in AMS XML Schemas

2.2.5. 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_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_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_RelationIsVersionOf	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 2. List of auto-generated elements supported by ACMA

2.2.6. Application Profiles

The ARCO consortium has been investigating the concept of application profiles. An application profile is defined as a schema that consists of data elements drawn from one or more namespaces and optimised for a particular local application [19].

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 and proposed guidelines for XML Schema implementers [20].

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.

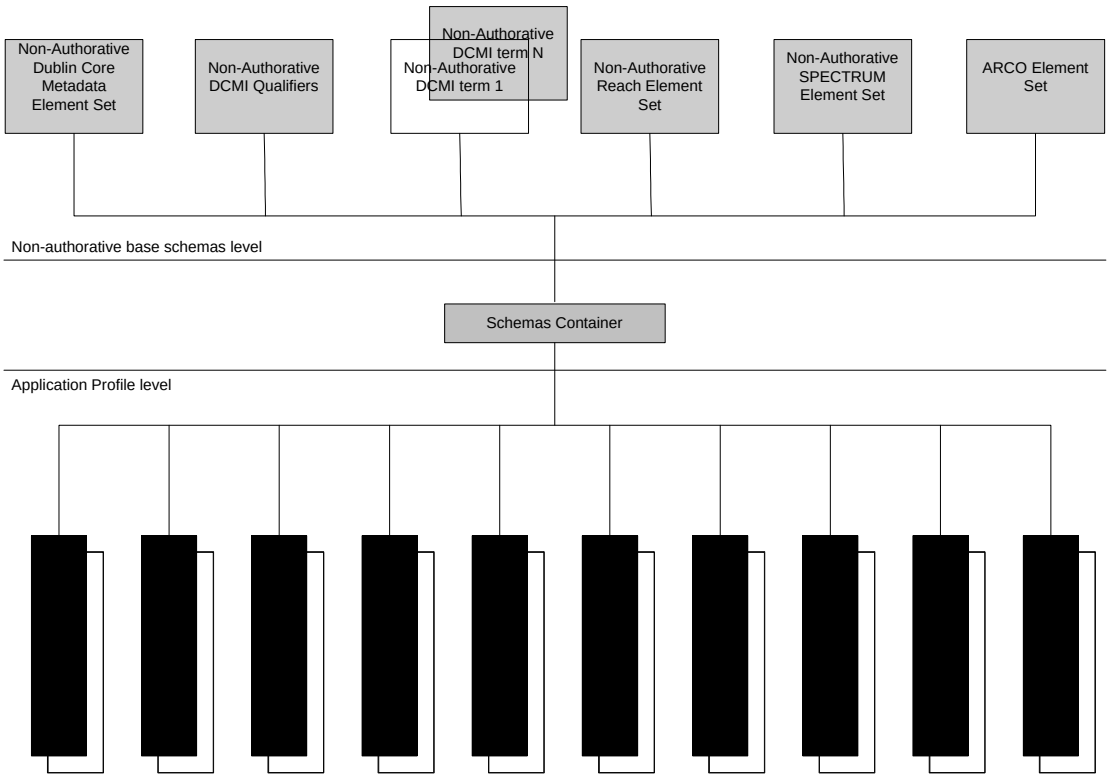


Figure 4. The architecture of application profiles in ARCO

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 27 elements. These elements are described below. For actual implementation, see Appendix B.

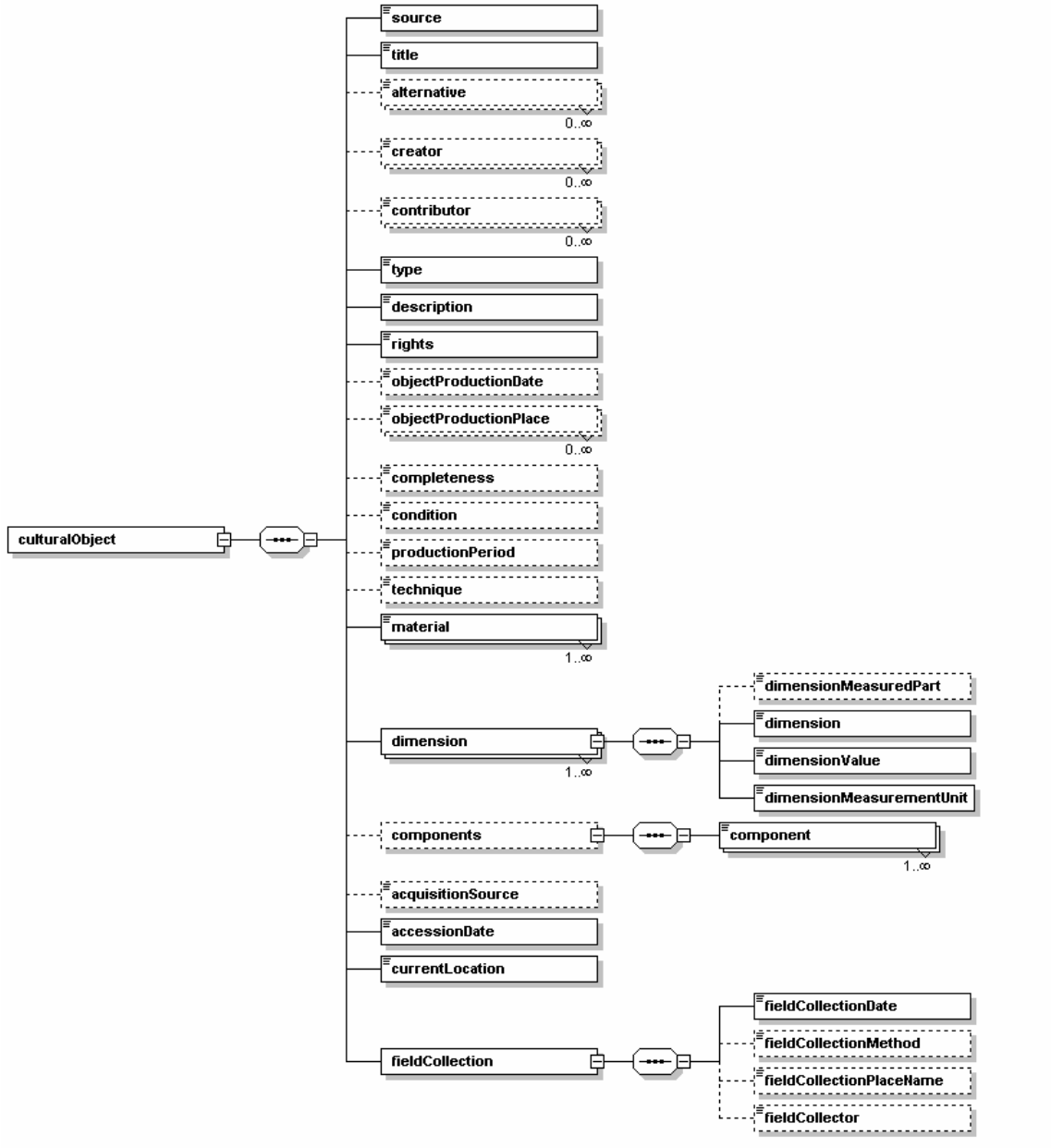


Figure 5. The overall XML Schema of the Cultural Object AMS

Element: source (CO Source)

The `source` element implements CO AMS element *Source* [1]. It defines 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`. In Figure 6, the XML Schema for the `source` element is presented.

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

Figure 6. XML Schema of the source element

Element: title (CO Name)

The `title` element implements CO AMS element *Name* [1]. It defines 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`and it is ascribed with intelligence properties. In Figure 7, the XML Schema for the `name` element is presented.

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

Figure 7. XML Schema of the name element

Element: alternative (CO Name Alternative)

The `alternative` element implements the CO AMS element *Name Alternative* [1]. It defines 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` and it is ascribed with intelligence properties (see Figure 8).

```
<xsd:element name="alternative" 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]. It defines 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` and it is ascribed with intelligence properties. The XML Schema for the `creator` element is presented in Figure 9.

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

Figure 9. XML Schema of the creator element

Element: contributor (CO Contributor)

The `contributor` element implements the CO AMS element *Contributor* [1]. It defines 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` and it is ascribed with intelligence properties. The XML Schema for the `contributor` element is presented in Figure 10.

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

Figure 10. XML Schema of the contributor element

Element: type (CO Type)

The `type` element implements the CO AMS element *Type* [1]. 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` and it is ascribed with intelligence properties (see Figure 11).

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

Figure 11. XML Schema of the type element

Element: description (CO Description)

The `description` element implements the CO AMS element *Description* [1]. It defines 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 12).

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

Figure 12. XML Schema of the description element

Element: rights (CO Rights)

The `rights` element implements the CO AMS element *Rights* [1]. It defines information about rights held in and over the cultural artefact. The `rights` 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 13).

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

Figure 13. XML Schema of the rights element

Element: objectProductionDate (CO Object Production Date)

The object`ProductionDate` element implements the CO AMS element *Object Production Date* [1]. It defines the date or period in which the cultural artefact was created. The object`ProductionDate` element is an optional child element of the `culturalObject` element, and it may occur only once. The type of the element is `xsd:string` and it is ascribed with intelligence properties. The XML Schema for the `dateCreated` element is presented in Figure 14.

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

Figure 14. XML Schema of the productionDate element

Element: objectProductionPlace (CO Production Place)

The object`ProductionPlace` element implements the CO AMS element *Object production place* [1]. It defines the geographical location in which an object was created. The `objectProductionPlace` element is an optional child element of the `culturalObject` element, and it may occur multiple times. The type of the element is `xsd:string` and it is ascribed with intelligence properties. (see Figure 15).

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

Figure 15. XML Schema for the productionPlace element

Element: completeness (CO Completeness)

The `completness` element implements the CO AMS element *Completeness* [1]. 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` and it is ascribed with intelligence properties. 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 16.

```
<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 16. XML Schema of the completeness element

Element: condition (CO Condition)

The `condition` element implements the CO AMS element *Condition* [1]. It defines 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` and it is ascribed with intelligence properties (see Figure 17).

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

Figure 17. XML Schema of the condition element

Element: productionPeriod (CO Production Period)

The `productionPeriod` element implements the CO AMS element *Production Period* [1]. It defines a broader example of date '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` and it is ascribed with intelligence properties. The XML Schema of the `productionPeriod` element is presented in Figure 18.

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

Figure 18. XML Schema of the productionPeriod element

Element: technique (CO Production Technique)

The `technique` element implements the CO AMS element *Production Technique* [1]. It defines the processes, methods, techniques and tools used to fabricate or decorate the object. The `productionTechnique` is an optional child element of the `culturalObject` element, and it may occur only once. The type of the element is `xsd:string` and it is ascribed with intelligence properties (see Figure 19).

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

Figure 19. XML Schema of the productionTechnique element

Element: material (CO Material)

The `material` element implements the CO AMS element *Material* [1]. It defines the basic material from which an object is constructed. The `material` element is a mandatory child element of the `culturalObject` element, and it may occur multiple times. The type of the element is `xsd:string` and it is ascribed with intelligence properties (see Figure 20).

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

Figure 20. XML Schema of the material element

Element: dimension

The dimension element serves implementation purposes. It is a complex type element defined by a sequence that groups all the dimension related CO AMS elements. The dimension element is an optional child element of the culturalObject element, and it may occur multiple times see Figure 21).

```
<xsd:element name="dimension" minOccurs="0" maxOccurs="unbounded">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="dimensionMeasuredPart" type="xsd:string" minOccurs="0"/>
         <xsd:element name="dimension" type="xsd:string" minOccurs="0"/>
         <xsd:element name="dimensionValue" type="xsd:string" minOccurs="0"/>
         <xsd:element name="dimensionMeasurementUnit" type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

Figure 21. XML Schema of the dimension element

Element: dimensionMeasuredPart (CO Dimension Measured Part)

The dimensionMeasuredPart element implements the CO AMS element *Dimension Measured Part* [1]. It defines the part of the object measured. The dimensionMeasuredPart element is an optional child element of the dimension element, and it may occur only once. The type of the element is xsd:string (see Figure 22).

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

Figure 22. XML Schema of the dimensionMeasuredPart element

Element: dimension (CO Dimension)

The dimension element implements the CO AMS element *Dimension* [1]. It defines the aspect of a part or component of an object being measured. The dimension element is a mandatorychild element of the dimension element, and it must occur only once. The type of the element is xsd:string (see Figure 23).

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

Figure 23. XML Schema of the dimension element

Element: dimensionValue (CO Dimension Value)

The dimensionValue element implements the CO AMS element *Dimension Value* [1]. It defines the numeric value of the measurement of a dimension. The dimensionValue element is amandatory child element of the culturalObject element, and it must occur only once. The type of the element is xsd:string (see Figure 24).

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

Figure 24. XML Schema of the dimensionValue element

Element: dimensionMeasurementUnit (CO Dimension Measurement Unit)

The dimensionMeasurementUnit element implements the CO AMS element *Dimension Measurement Unit* [1]. It defines the unit of measurement used when measuring a dimension. The dimensionMeasurementUnit element is amandatory child element of the culturalObject element, and it must occur only once. The type of the element is xsd:string (see Figure 25).

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

Figure 25. XML Schema of the dimensionMeasurementUnit element

Element: components (CO Components)

The `components` element implements the CO AMS element *Components* [1]. It defines a 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` component elements. This element should be used when the physical artefact consists of more than one component (see Figure 26).

```
<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 26. XML Schema of the components element

Element: aquisitionSource (CO Acquisition Source)

The `aquisitionSource` element implements the CO AMS element *Acquisition Source* [1]. It defines the organisation or person, or people from whom the object was obtained. The `aquisitionSource` element is an optional child element of the `culturalObject` element, and it may occur only once. The type of the element is `xsd:string` and it is ascribed with intelligence properties (see Figure 27).

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

Figure 27. XML Schema of the aquisitionSource element

Element: accessionDate (CO Accession Date)

The `accessionDate` element implements the CO AMS element *Accession Date* [1]. It defines the date on which an object formally enters the collections and is recorded into the accessions register. The `accessionDate` element is a mandatory child element of the `culturalObject` element, and it must occur only once. The type of the element is `xsd:string` and it is ascribed with intelligence properties (see Figure 28).

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

Figure 28. XML Schema of the accessionDate element

Element: currentLocation (CO Current Location)

The `currentLocation` element implements the CO AMS element *Current Location* [1]. It defines the physical place within an institution where an object is currently located. The `currentLocation` 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 29).

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

Figure 29. XML Schema of the currentLocation element

Element: fieldCollection

The `fieldCollection` element serves implementation purposes. It is a complex type element defined by a sequence that groups all the `fieldCollection related CO AMS elements`. The `fieldCollection` element is an optional child element of the `culturalObject` element, and it may occur multiple times (see Figure 30) .

```
<xsd:element name="fieldCollection">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="fieldCollectionDate" type="xsd:string"/>
         <xsd:element name="fieldCollectionMethod" type="xsd:string" minOccurs="0"/>
         <xsd:element name="fieldCollectionPlaceName" type="xsd:string" minOccurs="0"/>
         <xsd:element name="fieldCollector" type="xsd:string" minOccurs="0"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

Figure 30. XML Schema of the dimension element

Element: fieldCollectionDate (CO Field Collection Date)

The `fieldCollectionDate` element implements the CO AMS element *Field Collection Date* [1]. It defines the date an object was collected in the field. The `fieldCollectionDate` element is a mandatory child element of the `fieldCollection` element, and it must occur only once. The type of the element is `xsd:string` (see Figure 31).

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

Figure 31. XML Schema of the fieldCollectionDate element

Element: fieldCollectionMethod (CO Field Collection Method)

The `fieldCollectionMethod` element implements the CO AMS element *Field Collection Method* [1]. It defines the method used to excavate or collect an object in the field. The `fieldCollectionMethod` element is an optional child element of the `fieldCollection` element, and it may occur only once. The type of the element is `xsd:string` (see Figure 32).

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

Figure 32. XML Schema of the fieldCollectionMethod element

Element: fieldCollectionPlaceName (CO Field Collection Place)

The `fieldCollectionPlaceName` element implements the CO AMS element *Field Collection Place* [1]. It defines the name of the place where an object was collected in the field. The `fieldCollectionPlace` element is an optional child element of the `fieldCollection` element, and it may occur only once. The type of the element is `xsd:string` (see Figure 33).

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

Figure 33. XML Schema of the fieldCollectionPlace element

Element: fieldCollector (CO Field Collector)

The `fieldCollector` element implements the CO AMS element *Field Collector* [1]. It defines the name of the person or organisation responsible for collecting a specimen or object in the field. The `fieldCollector` element is an optional child element of the `fieldCollection` element, and it may occur only once. The type of the element is `xsd:string` (see Figure 34).

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

Figure 34. XML Schema of the fieldCollector 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 35. The schema

consists of 10 elements. These elements are described below. For actual implementation see Appendix C.

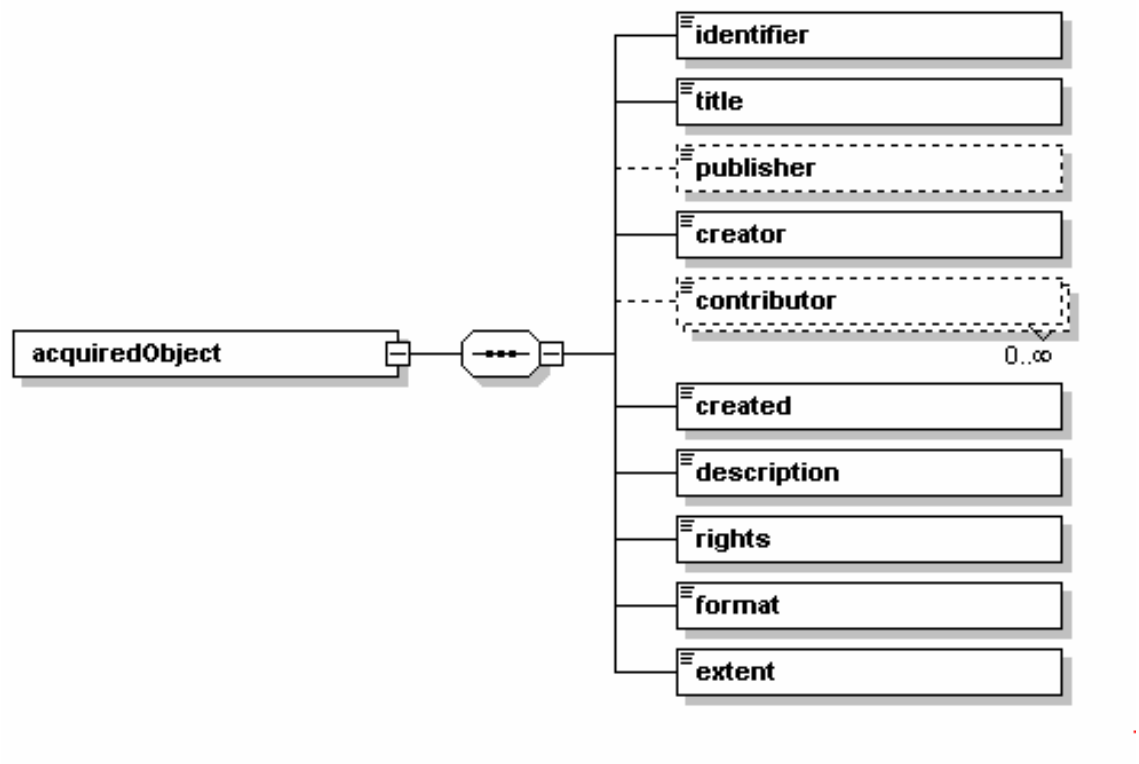


Figure 35. The overall XML Schema for Acquired Object AMS

Element: identifier (AO Identifier)

The `identifier` element implements the AO AMS element *Identifier* [1]. It defines 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`. 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 36.

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

Figure 36. XML Schema of the identifier element

Element: name (AO Name)

The `name` element implements the AO AMS element *Name* [1]. It defines 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 37). The element is auto-generated by the use of the `AO_Name` code name.

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

Figure 37. XML Schema of the title element

Element: publisher (AO Publisher)

The `publisher` element implements the AO AMS element *Publisher* [1]. It defines about the 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` (see Figure 38).

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

Figure 38. XML Schema of the publisher element

Element: creator (AO Creator)

The `creator` element implements the AO AMS element *Creator* [1]. It defines 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` (see Figure 39). 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" type="xsd:string"/>
```

Figure 39. XML Schema of the creator element

Element: contributor (AO Contributor)

The `contributor` element implements the AO AMS element *Contributor* [1]. It defines 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` (see Figure 40).

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

Figure 40. XML Schema of the contributor element

Element: created (AO Date Created)

The `created` element implements the AO AMS element *Date Created* [1]. It defines the date of creation of the Acquired Object. The `created` 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 element is auto-generated by ACMA with the `AO_DateCreated` code name. The XML Schema for the `dateCreated` element is presented in Figure 41.

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

Figure 41. XML Schema of the created element

Element: description (AO Description)

The `description` element implements the AO AMS element *Description* [1]. It defines 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 XML Schema for the `description` element is presented in Figure 42.

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

Figure 42. XML Schema of the description element

Element: rights (AO Rights)

The `rights` element implements the AO AMS element *Rights* [1]. It defines 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 43).

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

Figure 43. XML Schema of the rights element

Element: format (AO Format)

The `format` element implements the AO AMS element *Format* [1]. It defines 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` (see Figure 44). The `format` element is auto-generated by ACMA with the `AO_Format` code name.

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

Figure 44. XML Schema of the format element

Element: extent (AO Format Extent)

The `extent` element implements the AO AMS element *Format Extent* [1]. It defines the digital size of the Acquired Object. The `extent` 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` (see Figure 45). The `extent` element is auto-generated by ACMA with the `AO_FormatExtent` code name.

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

Figure 45. XML Schema of the extent 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 final 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 46. The schema consists of 11 elements. The `identifier`, `title`, `publisher`, `creator`, `contributor`, `created`, `description`, `rights`, `format`, and `formatExtent` are the same as in the AO AMS XML Schema. One new element: `relationIsVersionOf` is described below.

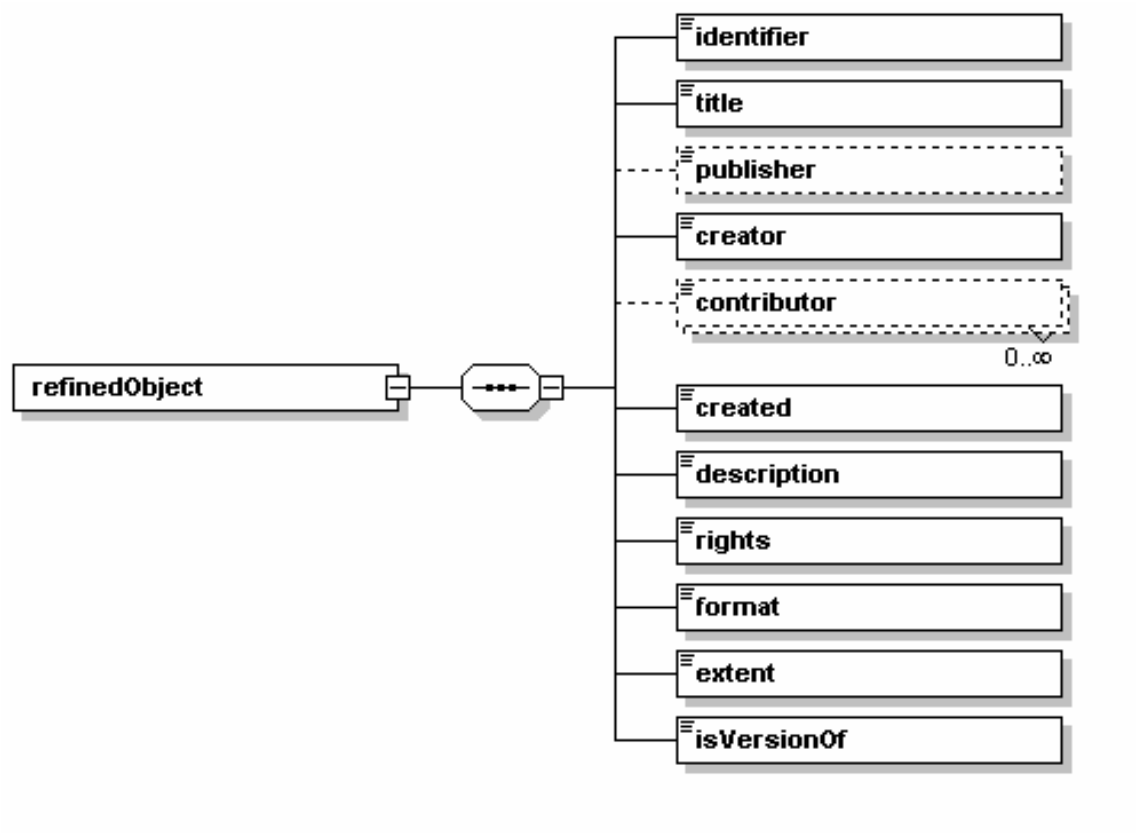


Figure 46. The overall XML Schema for Refined Object AMS

Element: isVersionOf (RO RelationIsVersionOf)

The `isVersionOf` element implements the RO AMS element *Relation Is Version Of* [1]. It defines an ordered list of identifiers of Cultural Objects (Acquired Objects and Refined Objects) on the refinement path. The `isVersionOf` element is a mandatory child element of the `refinedObject` element, and it must occur only once. The type of the element is `xsd:string`. The XML Schema for the `relationIsVersionOf` element is presented in Figure 47. The `isVersionOf` element is auto-generated by ACMA with the `RO_RelationIsVersionOf` code name.

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

Figure 47. XML Schema of the isVersionOf 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 48. The schema consists of 11 elements. These elements are described below. For actual implementation see Appendix E.

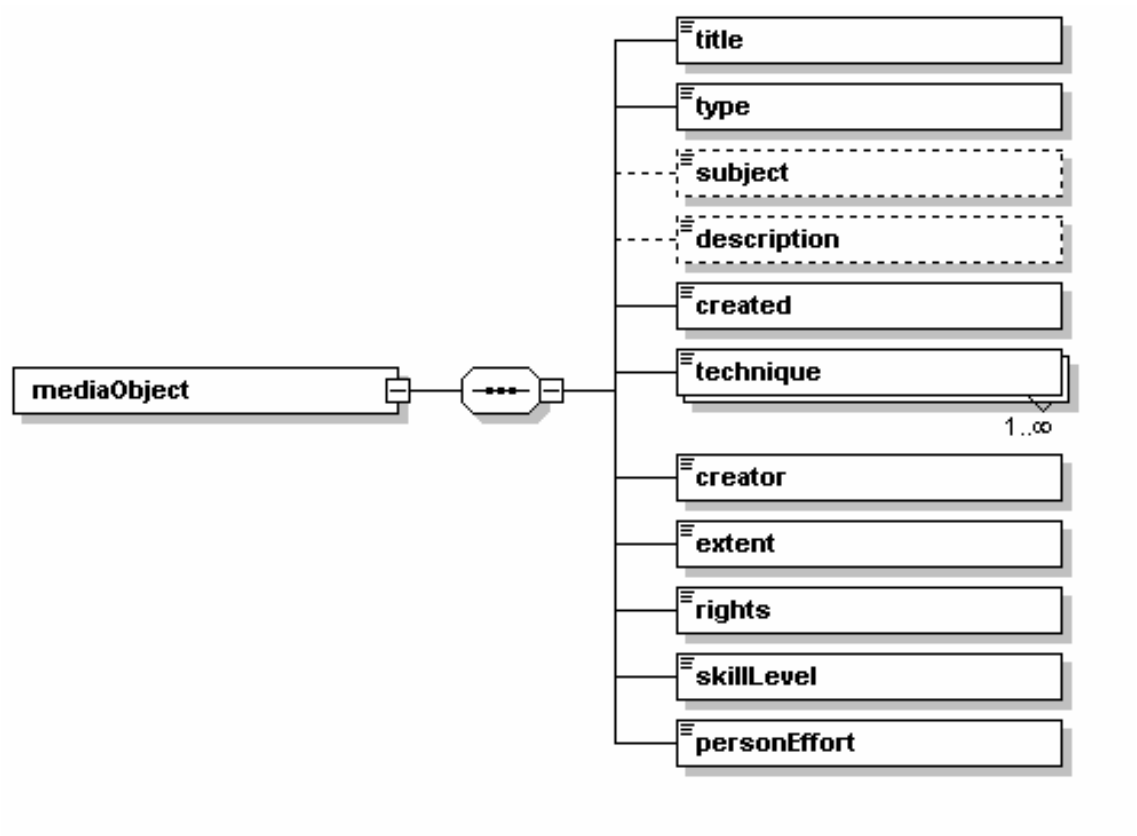


Figure 48. The overall XML Schema of Media Object AMS

Element: title (MO Name)

The `title` element implements the MO AMS element *Name* [1]. It defines the name of the Media Object. The `title` 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 49). The `title` element is auto-generated by ACMA with the code name `MO_Name`.

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

Figure 49. XML Schema of the title element

Element: type (MO Type)

The `type` element implements the MO AMS element *Type* [1]. It defines a list of mime-types for complex Media Object or a single mime-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 `xsd:string` (see Figure 50). The `type` element is auto-generated by ACMA with the code name `MO_Type`.

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

Figure 50. XML Schema of the type element

Element: subject (MO Subject)

The `subject` element implements the MO AMS element *Subject* [1]. The *subject* element defines 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 may occur only once. The type of the element is `xsd:string` (see Figure 51).

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

Figure 51. XML Schema of the subject element

Element: description (MO Description)

The `description` element implements the MO AMS element *Description* [1]. It defines 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` and it is ascribed with effort report properties (see Figure 52).

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

Figure 52. XML Schema of the description element

Element: created (MO Date Created)

The `created` element implements the MO AMS element *Date Created* [1]. It defines the date of creation of the Media Object. The `created` element is a mandatory child element of the `mediaObject` element, and it must occur only once. The type of the element is `xsd:dateTime` (see Figure 53). The `created` element is auto-generated by ACMA with the code name `MO_DateCreated`.

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

Figure 53. XML Schema of the created element

Element: technique (MO Technique)

The `technique` element implements the MO AMS element *Technique* [1]. It defines the technique used to acquire a digital manifestation of the Cultural Object. The `technique` element is a mandatory child element of the `mediaObject` element, and it may occur multiple times. The type of the element is `xsd:string` and it is ascribed with effort report properties (see Figure 54).

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

Figure 54. XML Schema of the technique element

Element: creator (MO Creator)

The `creator` element implements the MO AMS element *Creator* [1]. It defines 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 must occur only once. The type of the element is a list of `xsd:string` and it is ascribed with effort report properties (see Figure 55). 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" type="xsd:string"/>
```

Figure 55. XML Schema of the creator element

Element: extent (MO Format Extent)

The `extent` element implements the MO AMS element *Format Extent* [1]. It defines the digital size of the Media Object. The `extent` element is a mandatory child element of the `mediaObject` element, and it must occur only once. The type of the element is based on `xsd:string` (see Figure 56). The `extent` element is auto-generated by ACMA with the `MO_FormatExtent` code name.

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

Figure 56. XML Schema of the extent element

Element: rights (MO Rights)

The `rights` element implements the MO AMS element *Rights* [1] . It defines 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 57).

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

Figure 57. XML Schema of the rights element

Element: skillLevel (MO Skill Level)

The `skillLevel` element implements the MO AMS element *Skill Level* [1] . It defines information about the skill level or experience of the creator of the Media Object. The `skillLevel` element is a mandatory child element of the `mediaObject` element and should occur once. The type of the element is `xsd:string` and it is ascribed with effort report properties (see Figure 58).

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

Figure 58. XML Schema of the skillLevel element

Element: personEffort (MO Person Effort)

The `personEffort` element implements the MO AMS element *Person Effort* [1] . It defines information about the amount of time spent in capturing the Media Object. The `personEffort` element is a mandatory child element of the `mediaObject` element, and it must occur only once. The type of the element is `xsd:decimal` and it is ascribed with effort report properties (see Figure 59).

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

Figure 59. XML Schema of the personEffort 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 60. The schema consists of five elements. These elements are described below. For actual implementation see Appendix F.

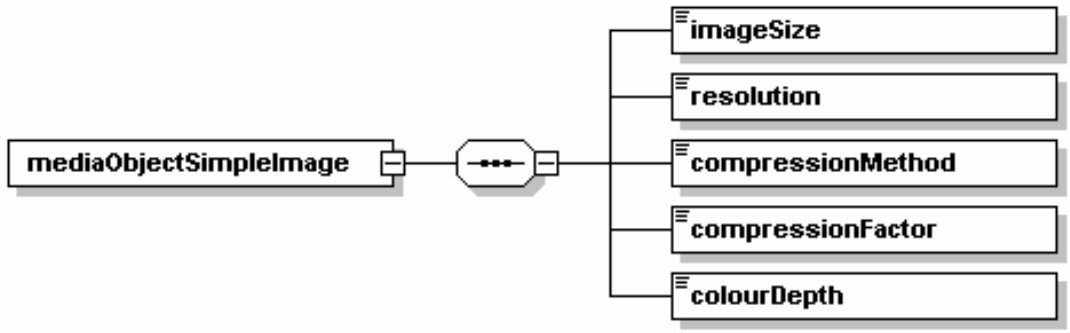


Figure 60. The overall XML Schema for Simple Image AMS

Element: imageSize (MOT Simple Image: Image Size)

The `imageSize` element implements the MOT Simple Image AMS element *Image Size* [1]. It defines the Image width and height in pixels. 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` (see Figure 61). The `imageSize` element is auto-generated by ACMA with the code name `MOT_SimpleImage_ImageSize`.

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

Figure 61. XML Schema of the imageSize element

Element: resolution (MOT Simple Image: Resolution)

The `resolution` element implements the MOT Simple Image AMS element *Resolution* [1]. It defines 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 62). The `resolution` element is auto-generated by ACMA with the code name `MOT_SimpleImage_Resolution`.

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

Figure 62. 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]. It defines 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` (see Figure 63). The element is auto-generated by ACMA with the code name `MOT_SimpleImage_CompressionMethod`.

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

Figure 63. 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]. It defines 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`. (see Figure 64).

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

Figure 64. 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]. It defines 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 65). The element is auto-generated by ACMA with the code name `MOT_SimpleImage_ColourDepth`.

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

Figure 65. 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 66. The schema consists of two elements. These elements are described below. For actual implementation see Appendix G.

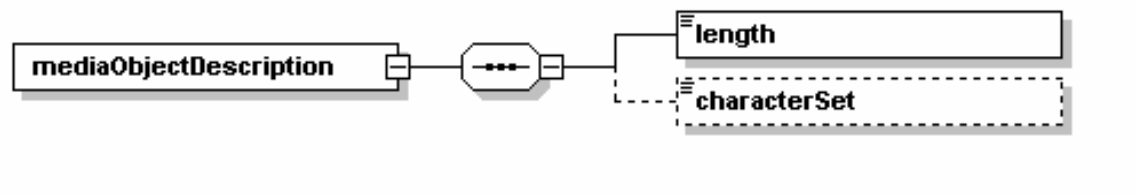


Figure 66. The overall XML Schema for Description AMS

Element: length (MOT Description: Length)

The `length` element implements the MOT Description AMS element *Length* [1]. It defines 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 67). The element is auto-generated by ACMA with the code name `MOT_Description_Length`.

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

Figure 67. XML Schema of the length element

Element: characterSet (MOT Description: Character set)

The `characterSet` element implements the MOT Description AMS element *Character set* [1]. It defines 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 68. The schema consists of two elements. These elements are described below. For actual implementation see Appendix H.

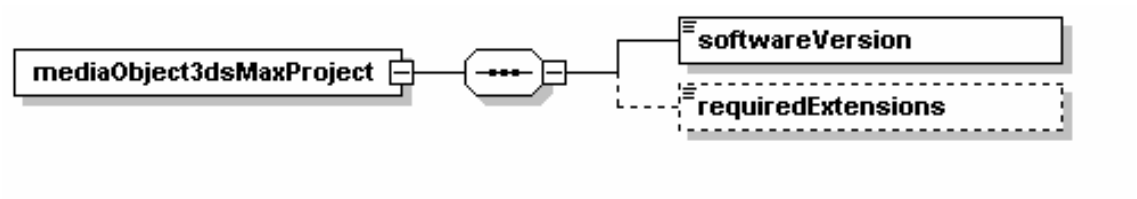


Figure 68. The overall XML Schema for 3D Studio Max AMS

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

The `softwareVersion` element implements the MOT 3D Studio Max Project AMS element *Software version* [1]. It defines 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 `xsd:string` (see Figure 69).

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

Figure 69. XML Schema of the softwareVersion element

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

The `requiredExtensions` element implements the MOT 3D Studio Max Project AMS element required extensions [1]. It defines extensions and plug-ins needed to properly open and display this file. The `requiredExtensions` element is an optional child element of the `mediaObject3dsMaxProject` element, and it may occur multiple times. The type of the element is `xsd:string` (see Figure 70).

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

Figure 70. 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 71. The schema consists of four elements. These elements are described below. For actual implementation see Appendix I.

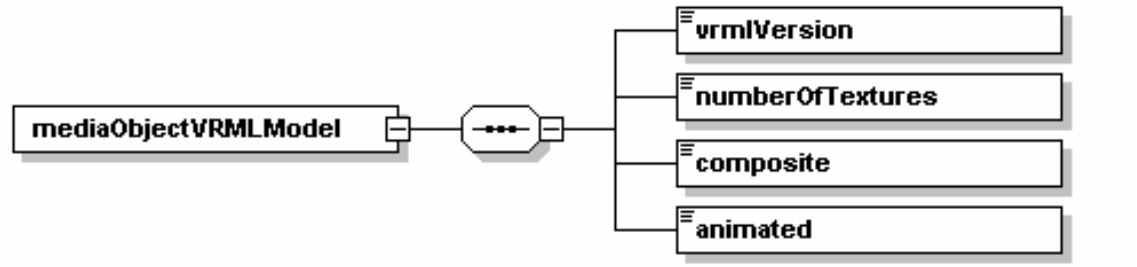


Figure 71. The overall XML Schema for VRML Model AMS

Element: vrmlVersion (MOT VRML Model: VRML version)

The `vrmlVersion` element implements the MOT VRML Model AMS element *VRML version* [1]. It defines 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` (see Figure 72). The `vrmlVersion` element is auto-generated by ACMA with the code name `MOT_VRML_Version`.

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

Figure 72. 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]. It defines 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 73). The element is auto-generated by ACMA with the code name `MOT_VRML_NumberOfTextures`.

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

Figure 73. XML Schema of the numberOfTextures element

Element: composite (MOT VRML Model: Composite)

The `composite` element implements the MOT VRML Model AMS element *Composite* [1]. It defines 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 74). The element is auto-generated by ACMA with the code name `MOT_VRML_Composite`.

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

Figure 74. XML Schema of the composite element

Element: animated (MOT VRML Model: Animated)

The `animated` element implements the MOT VRML Model AMS element *Animated* [1]. It defines 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 75). The element is auto-generated by ACMA with the code name `MOT_VRML_Animated`.

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

Figure 75. 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 76. The schema consists of two elements. These elements are described below. For actual implementation see Appendix J.

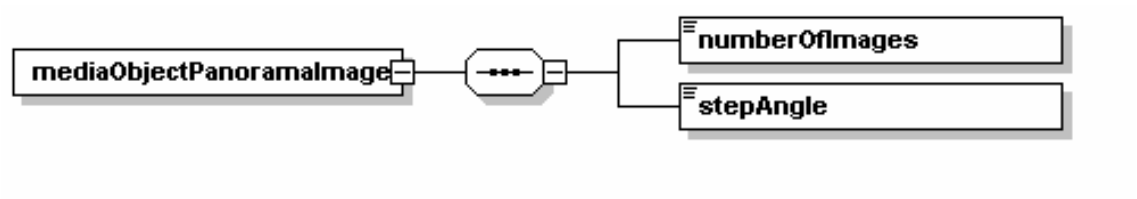


Figure 76. The overall XML Schema for Panorama Image AMS

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

The `numberOfImages` element implements the MOT Panorama Image AMS element *Number of images* [1]. It defines 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 77). The element is auto-generated by ACMA with the code name `MOT_PanoramaImage_NumberOfImages`.

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

Figure 77. 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]. It defines 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 78). The element is auto-generated by ACMA with the code name `MOT_PanoramaImage_StepAngle`.

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

Figure 78. 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 79. The schema consists of three elements. These elements are described below. For actual implementation see Appendix K.

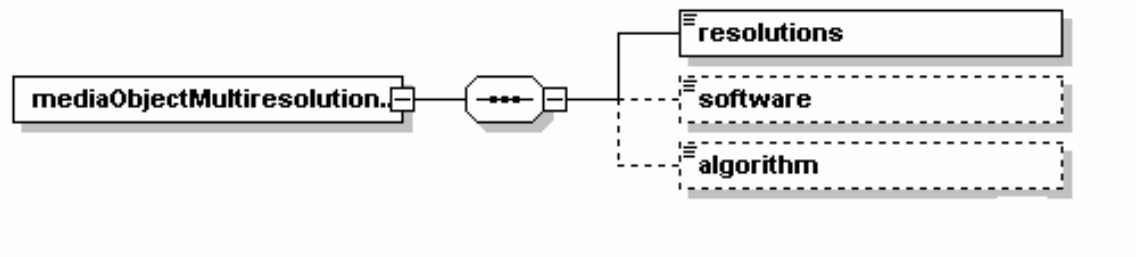


Figure 79. The overall XML Schema for Multiresolution Image AMS

Element: resolutions (MOT Multiresolution Image: Resolutions)

The `resolutions` element implements the MOT Multiresolution Image AMS element *Resolutions* [1]. It defines 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` (see Figure 80). The `resolutions` element is auto-generated by ACMA with the code name `MOT_MultiresolutionImage_Resolutions`.

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

Figure 80. XML Schema of the resolutions element

Element: software (MOT Multiresolution Image: Software)

The `software` element implements the MOT Multiresolution Image AMS element *Software* [1]. It defines 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 81).

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

Figure 81. XML Schema of the software element

Element: algorithm (MOT Multiresolution Image: Algorithm)

The `algorithm` element implements the MOT Multiresolution Image AMS element *Algorithm* [1]. It defines 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 82).

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

Figure 82. XML Schema of the algorithm element

2.4. AMS 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.

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.

2.4.1. Cultural Object Schemas

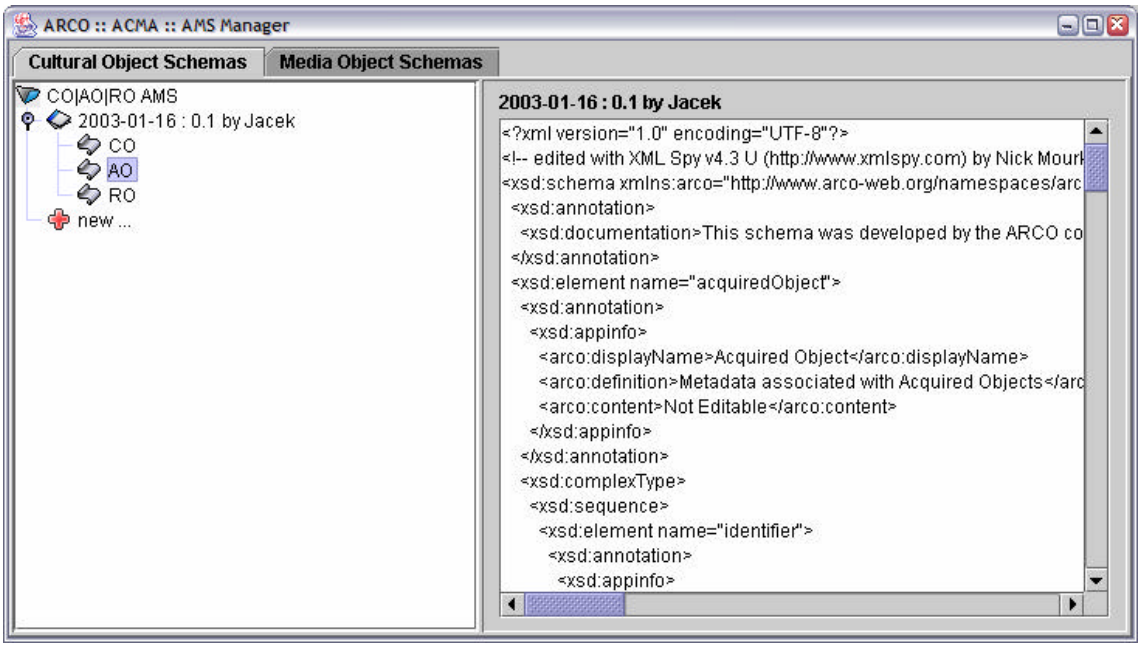


Figure 83. AMS Manager – Cultural Object Schemas

A screenshot of the Cultural Object panel of the AMS Schema Manager is shown in Figure 83. The left part of the window shows a time-ordered list of CO AMS schema versions. Each AMS schema version consists of:

- a version of the Cultural Object AMS sub-schema (CO AMS);
- a version of the Acquired Object AMS sub-schema (AO AMS);
- a version of the Refined Object AMS sub-schema (RO AMS).

These versions must be maintained together because each AMS version must consist of a consistent set of AMS sub-schema versions.

The AMS versions displayed in the left tree are described by their creation date followed by their version name. Every AMS version has three sub-nodes corresponding to: CO AMS, AO AMS, and RO AMS. By selecting a leaf node a user may view the corresponding AMS sub-schema in the right panel.

By selecting a special '*new…*' element at the end of the list, the user may create a new AMS version in the database. After selecting this element, a special form is displayed on the right side of the window. By the use of this form, a user may enter the new version name, and select the XML Schema files (*.xsd) from the file system (one for CO, one for AO, and one for RO), or indicate that this particular sub-schema should be copied from previous version of the AMS schema (checkbox on the left).

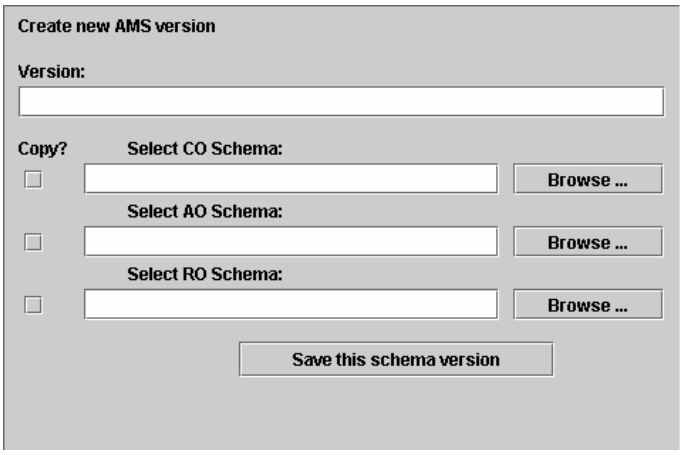


Figure 84. Adding new AMS schema in the ARCO AMS Schema Manager

An AMS schema version may be deleted from the system by selecting the delete item available in the right-click menu. Only unused AMS versions can be deleted from the database. 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.

Additionally each schema document may be saved to a file by selecting the *Save to File* item from right-click menu.

2.4.2. Media Object Schemas

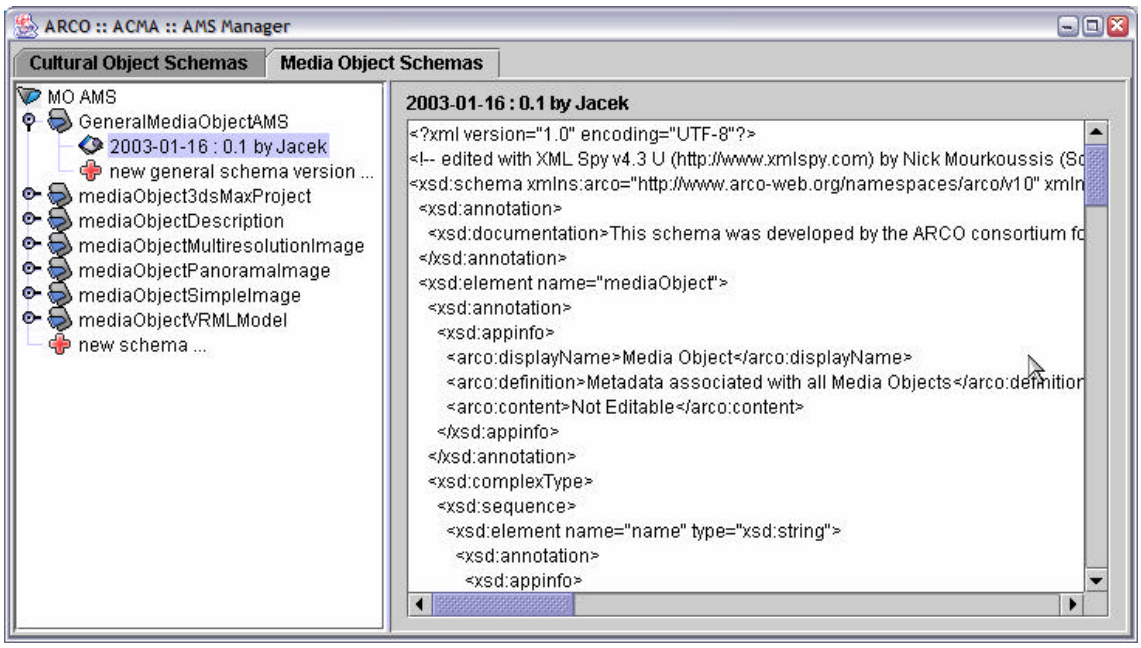


Figure 85. AMS Manager – Media Object Schemas

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 85. 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 ('MO 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…*' nodes, a user may add a new schema or a new schema version.

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,
- save a schema version to a file – by the use of the '*save to file*' option accessible in the right-click popup menu, or
- delete a schema or a schema version – by the use of the '*delete*' option accessible in the right-click popup menu.

When the "*new schema…*" operation is selected, a special form is displayed. The form allows the user to enter a new schema name and description.

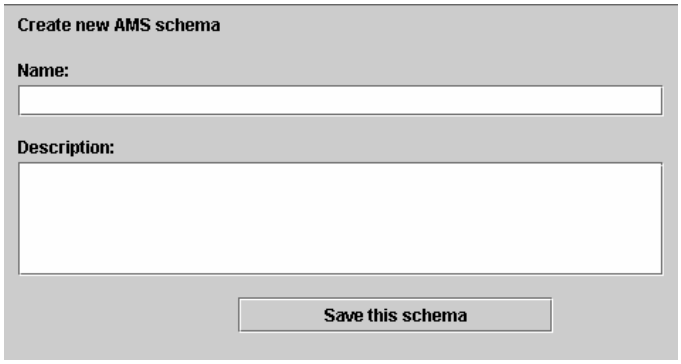


Figure 86. Creating a new AMS schema in MO AMS Manager

When the "*new version…*" operation is selected, another form is displayed that allows entering the version name and select the XML Schema file (*.xsd) containing new schema definition.

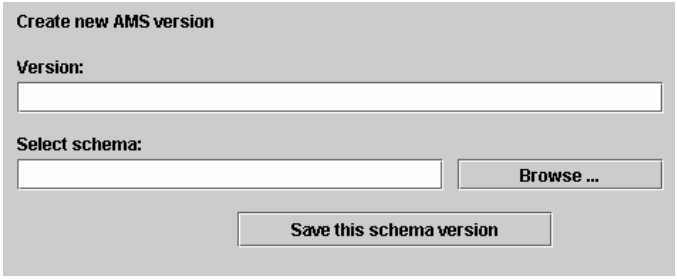


Figure 87. Creating a new version of AMS schema in MO AMS Manager

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 88. 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.

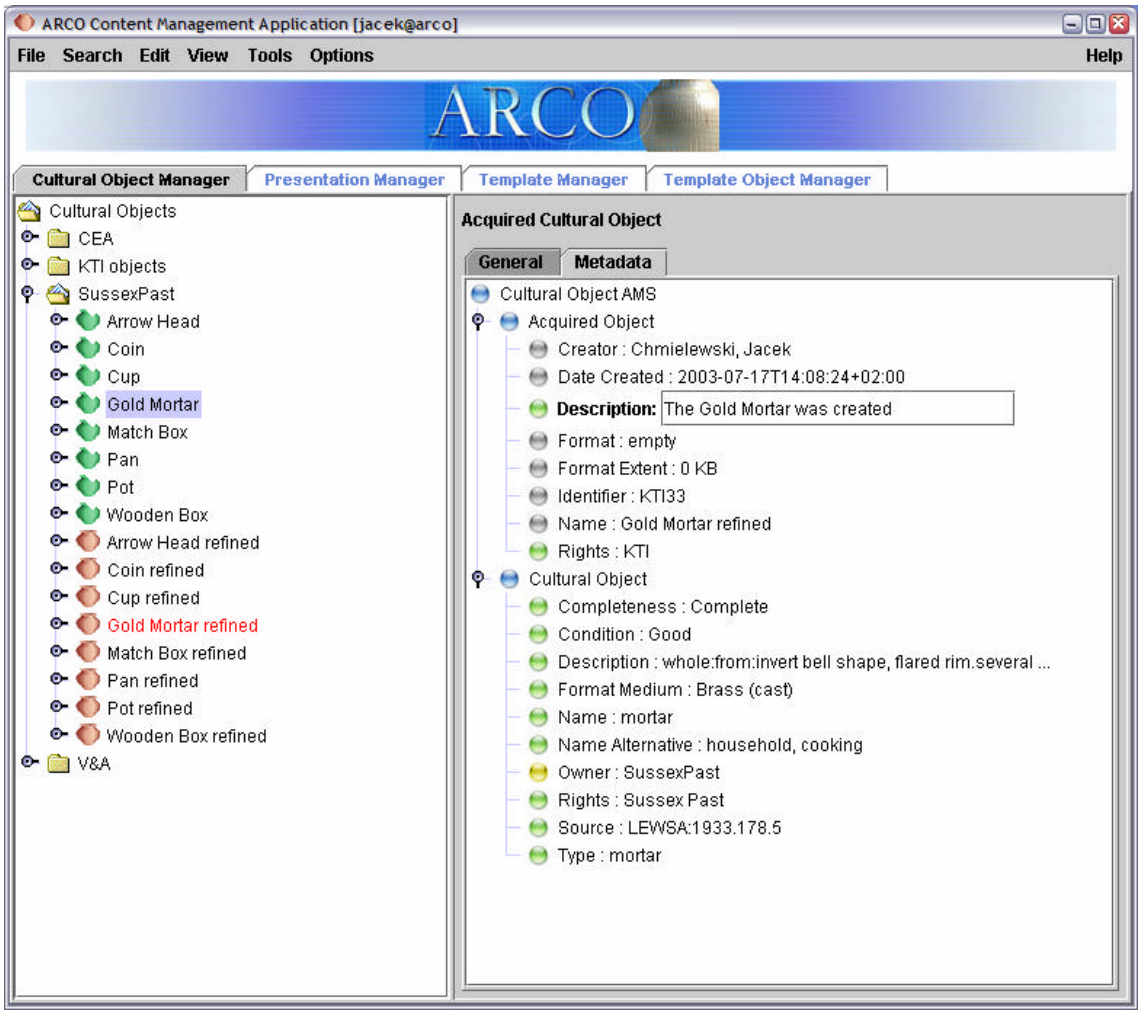


Figure 88. ARCO AMS Metadata Editor

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 89). 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.

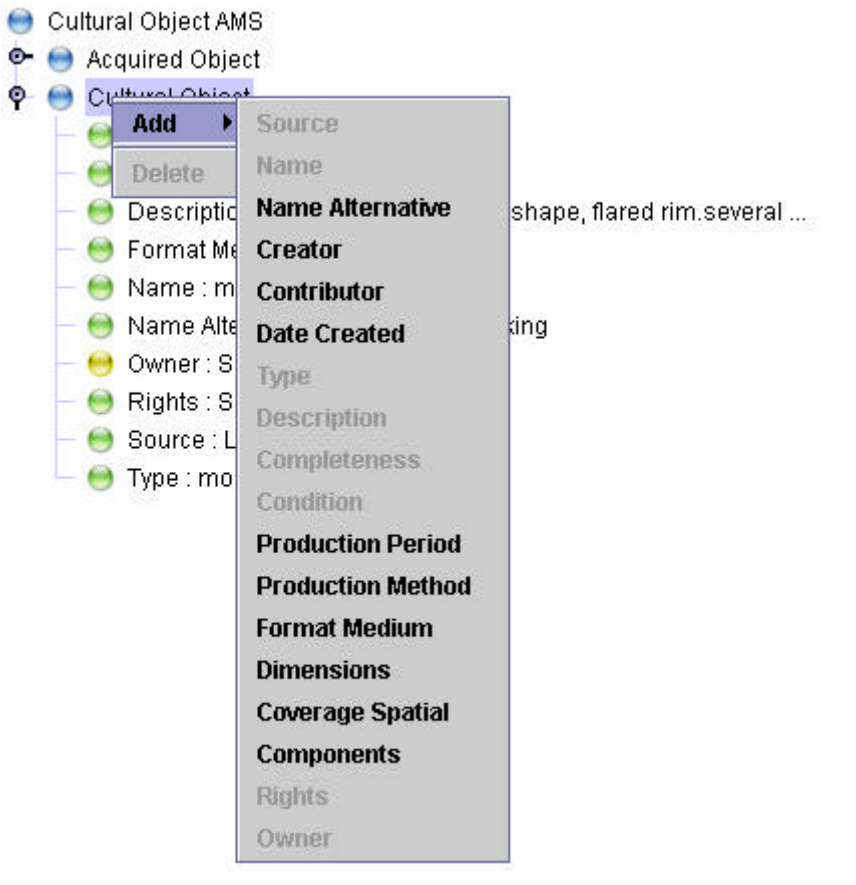


Figure 89. Adding a new element in AMS Metadata Editor

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 directly in the tree using a special editor for each node (available on double-click). The editor type depends on element definition. It might be a simple text field, drop-down menu with enumerated predefined values, or a dialog that allows choosing a value from a dictionary. When entered, the value is validated using AMS definition, and if it is not correct, a dialog window with appropriate error message appears.

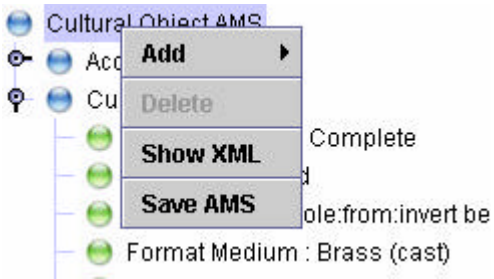


Figure 90. Additional functions of the AMS Metadata Editor

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 to an XML file (Figure 90).

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

	Blue	Complex elements that may contain other elements, but no values

	Green	Simple leaf element – may contain a value
	Grey	Auto-generated element (read-only) – value is generated by the application
	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 91.

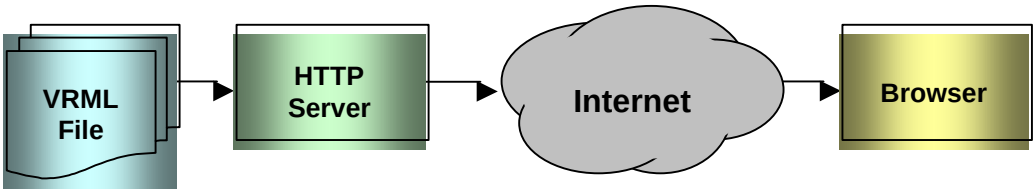


Figure 91. Architecture of a conventional VRML system

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 [6][7]. A solution proposed in [8][9][10] 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 uses, adapts and further extends a novel virtual reality modelling language called X-VRML. The X-VRML language has been designed by one the project partners [11][13][14][15][16] to enable dynamic modelling of virtual reality [12]. 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 with server-side model interpretation is presented in Figure 92. 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.

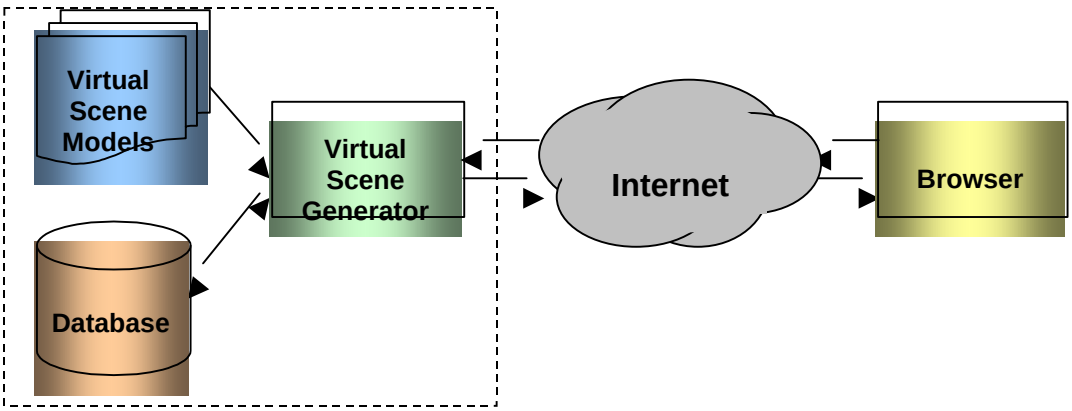


Figure 92. Architecture of a server-side model-based VR system

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.

In Figure 93, architecture of a model-based virtual reality system with client-side model interpretation is presented. Client-side interpretation permits to control the state of the virtual scene during the whole virtual scene life-time. This allows to fully exploit the X-VRML language features such as dynamic manipulation of the scene contents, handling user-interaction, implementation of persistency, etc. To allow client-side interpretation of X-VRML a special browser or an extension to a standard VRML browser is required. Such extension may be implemented as an applet connected to the VRML browser.

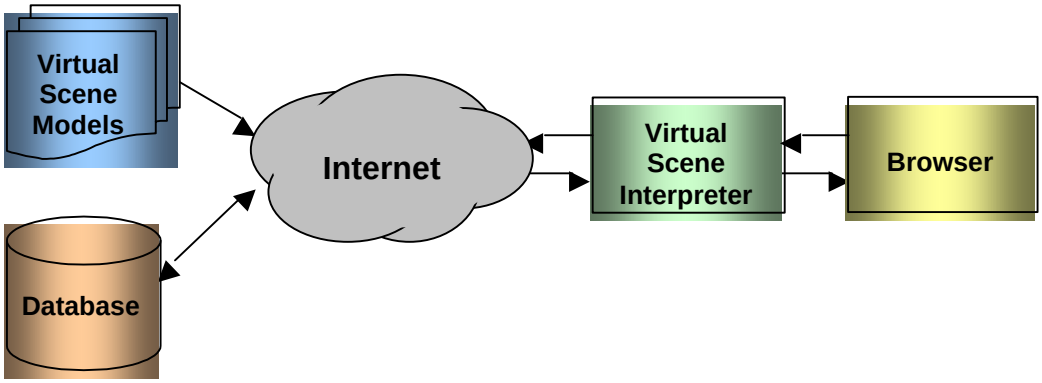


Figure 93. Architecture of a client-side model-based VR system

By the introduction of X-VRML, entirely new methods of designing virtual reality applications become possible. Dynamic modelling capabilities in combination with the possibility of retrieving any part of the virtual scene model from a database enable implementation of a system where both data and virtual world structure are modelled in a database. Direct advantages of such approach include possibility of maintaining multiple virtual scene models, possibility of automatically updating data in the virtual world model, possibility of enforcing consistency between elements of the virtual world model such as virtual scene models and virtual objects, possibility of maintaining efficiently large shared libraries of components, possibility of using different methods of visualization of the same data, and possibility of storing specifications of virtual scene model parameters and the parameters values for later reconstruction of virtual scenes created from the virtual world model.

All this features, accompanied by standard advantages of using databases, like consistency, local and remote access, multi-user access, powerful indexing and retrieval of contents, etc., greatly improve the overall functionality of system based on the dynamic modelling technique.

In Figure 94, an overall architecture of a system with all components of the virtual world model stored in a database is presented.

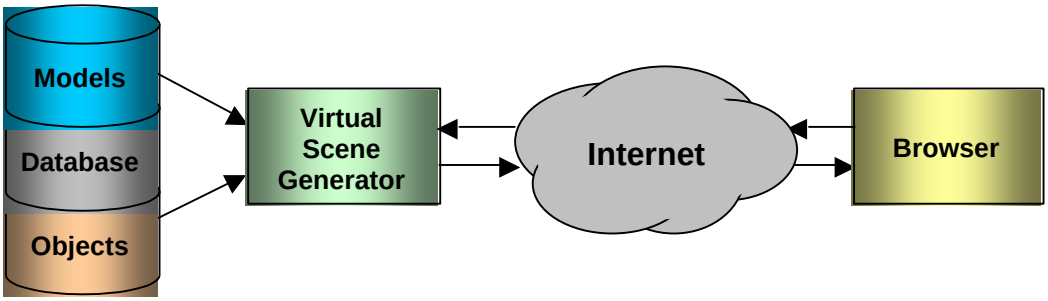


Figure 94. Architecture of a model-based system with all model elements stored in a database

3.2. Language Overview

The X-VRML language is based on XML [17]. 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 [11], and `Class/Instance` representing classes and their instances [13]. 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 `<DBQuery> ... </DBQuery>`. 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 3-1.

Java Version	X-VRML Version

`for (int i = 0; i <= 10; i++) {`
`  int x = i*i;`
`  if (i < 5)`
`  {`
`  ...`
`  }`
`  else`
`  {`
`  ...`
`  }`
`}`
	`<For name="i" from="0" to="10" step="1">`
	`  <Set name="x" value="{$i*$i}"/>`
	`  <If condition="{$i<5}">`
	`  <Then>`
	`  ...`
	`  </Then>`
	`  <Else>`
	`  ...`
	`  </Else>`
	`  </If>`
	`</For>`

Table 3-1. Comparison of Java and X-VRML versions of the same algorithm

3.2.1. X-VRML Modules

Functionality of the X-VRML language is divided into modules. Three modules of X-VRML are commonly used in both server-side and client-side systems: the Core Module, the Object-Oriented Module, and the Database Module. The Dynamic Module is used in client-side X-VRML systems.

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.

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

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

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, models enabling user interaction and models of persistent virtual scenes. The X-VRML Dynamic Module in connection with the Database Module is used in the final ARCO prototype to implement persistent dynamic scenes. Such scenes enable a museum curator to visually design virtual exhibitions, e.g. distribute cultural objects in a 3D exhibition gallery.

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, presentation properties).

3.2.2. X-VRML Core Module

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.

Variables

The X-VRML language allows using variables. Variables can be set inside X-VRML programs by the use of X-VRML commands (e.g., `Set`, `For`, `Iteration`, or `DBQuery`), 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).

Expressions

In X-VRML, all values of element attributes may contain expressions, which are evaluated prior to element interpretation. Expressions are contained in curly brackets – { and }. One attribute value specification may contain several distinct expressions, for example an X-VRML command:

```
<set name="obj{$i}" value="Object: {@name} volume={$x*$y*$z}"/>
```

will assign to variable *obj1* value *Object: Mortar volume=24* assuming that i=1, name=Mortar, x=2, y=3, and z=4.

The fact that expression boundaries are explicitly indicated by the curly brackets speeds up processing of attribute values since only selected parts are evaluated and simplifies syntax for constant attribute values that do not contain expressions.

X-VRML expressions can contain:

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

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.

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).

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.

Switch Statement

Switch statements are expressed by `Switch` elements and allow dynamically choosing one child-element that should be parsed. The syntax of the `Switch` element is as follows:

```
<Switch value="…">
      <Case values=”...,...,...”>
          ...
      </Case>
      <CaseRange from=”...” to=”...”>
          ...
      </CaseRange>
      <Default>
          ...
      </Default>
```

```
</Switch>
```

The `Switch` tag has one attribute: `value`. This mandatory attribute specifies an expression whose value is calculated and used to select the child element to be parsed.

There are three types of elements that can appear inside the Switch element: `Case`, `CaseRange` and `Default`.

The `Case` element has one attribute: `values` that contains a list of expressions. If the value provided in the `Switch` element is equal to any of the values in the case value list the `Case` element is parsed.

The `CaseRange` element has two parameters: `from` and `to`. If the value provided in the switch element is between the values provided in these parameters the `CaseRange` element is parsed.

The `Default` element is parsed if none of the `Case` or `CaseRange` elements were parsed in the current `Switch` element.

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.

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 `Inline`.

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 `Inline` 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 `Inline` 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 `Inline` element is as follows:

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

The `Inline` 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.

3.2.3. The X-VRML Database Module

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.

Connecting to Databases

In X-VRML, a connection to a database is established by the `DBConnect` element. The required connection details are provided in two parameters: `connString` and `connDriver`. 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 `DBConnect` element is as follows:

```
<DBConnect connString="..." connDriver="...">
...
</DBConnect>
```

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

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 `DBQuery` element. To simplify the process of retrieving data and constructing the output scene description, the `DBQuery` 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 `DBQuery` element is the following:

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

The `DBQuery` 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 `DBConnect` element. A single connection established by one `DBConnect` element may be used by more than one `DBQuery` element.

The `DBQuery` 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.

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 `DBUpdate` element. The syntax of this element is the following:

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

The `DBUpdate` 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 `DBUpdate` element must be located inside a `DBConnect` element.

3.2.4. X-VRML Object-Oriented Module

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.

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>
    <AttrDecl name="…" defValue="…"/>
    ...
  </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:

```
<AttrDecl name="…" defValue="…"/>
```

The `AttrDecl` 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,
- `defValue` 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="…" className="…">
...
</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 `className` 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 className="…"/>
```

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: `className` 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 `className` attribute. In case of multiple inheritance, where there are a number of superclasses for the class, the `className` 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).

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.

3.2.5. X-VRML Dynamic Module

Overview

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 result from the dynamism coded in the scene model, from the user interaction, or data read from a database.

As opposed to static X-VRML, processing of a dynamic X-VRML scene model is not terminated when the scene is sent to the browser. This implies that processing of the dynamic X-VRML models must be performed (at least partially) on the client site.

In typical dynamic X-VRML implementations, parsing of the X-VRML scene model is performed in two stages:

- initial parsing and interpretation of the scene that builds the initial scene structure;
- continuous interpretation of parts of the X-VRML model synchronized with events in the virtual scene. Continuous interpretation can result in manipulation of the scene contents.

Below, example commands of the dynamic X-VRML module are presented.

Synchronization with events

An important feature of the X-VRML language is possibility of implementing reaction to events generated in the virtual scene. An example element performing this task is `Listen`. The syntax of the `Listen` element is the following:

```
<Listen nodeName="..." eventName="..." varName="...">
...
</Listen>
```

The `Listen` element has two mandatory attributes: `nodeName` and `eventName` representing the name of the node that generates events, and the name of the event. The optional `varName` attribute provides name of X-VRML variable that will contain value of the event. The contents of the `Listen` element is interpreted each time the corresponding event is generated.

Sending events

Dynamic X-VRML processor may also send events to the virtual scene. Sending events is performed by the `SendEvent` command. The `SendEvent` element has the following syntax:

```
<SendEvent nodeName="..." eventName="..." value="..." number="..."/>
```

The mandatory `nodeName` and `eventName` attributes provide the name of the node and name of the event, respectively. The `value` attribute provides the value to be sent. Optional `number` attribute is used in case of multiple-value event types.

Delayed processing of a model fragment

In dynamic X-VRML scene models, some of the scene fragments can be interpreted and displayed with a delay synchronized with events in the virtual scene. To achieve this goal a special element `ShowAt` has been introduced. The syntax of the `ShowAt` node is the following:

```
<ShowAt nodeName="..." eventName="..." varName="...">
...
</ShowAt>
```

The interpretation of the `ShowAt` element (and its contents) is synchronized with events in the virtual scene. The event is identified by the `nodeName` and `eventName` attributes. Every time the event occurs in the virtual scene the `ShowAt` element is interpreted and resulting scene elements are added to the scene at the same time replacing elements added during the previous interpretation the element.

3.2.6. X-VRML ARCO Module

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 either the ARCO database or XDE files. The data retrieved includes exhibition spaces, Cultural Objects, Media Objects, AMS metadata, and visualization properties.

In case of retrieving data from the ARCO database 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, the 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. Moreover, the X-VRML ARCO module handles transparently data stored in XDE files. The same model can be used to present data stored in a database and in an XDE file when appropriate connection parameters are provided.

Information about ARCO objects such as exhibition spaces, Cultural Objects, Media Objects, are retrieved by the use of a set of X-VRML commands. Instead of creating a specialized element for each type of information, elements permitting retrieval of all types of information for a particular object type were created. The following elements are implemented in the final ARCO prototype:

- `<ARCO_AFProps>` for retrieval of data related to ARIF exhibition spaces,
- `<ARCO_COProps>` for retrieval of data related to Cultural Objects,
- `<ARCO_MOProps>` for retrieval of data related to Media Objects, and
- `<ARCO_VisProp>` for retrieval of visualization properties of ARIF exhibition spaces, Cultural Objects and Media Objects.

AMS metadata for Cultural Objects and Media Objects can be retrieved by the use of:

- `<ARCO_GetAMS>` command.

Another set of X-VRML commands is used for model parameterisation. These commands are used both by the ACMA tool for template parameters specification and by the ARIF for creation of HTML forms, e.g. used on the 'Search' pages. This set of X-VRML commands consists of the following elements:

- `<ARCO_InputText>` for definition of a text field input element,
- `<ARCO_InputCheckbox>` for definition of a checkbox,
- `<ARCO_InputSelect>` for definition of the multi-option selection element (like a combobox or a radio button).

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

Retrieving Data Related to ARIF Exhibition Spaces

To retrieve data related to ARIF exhibition spaces the `ARCO_AFProps` command is used. The `ARCO_AFProps` element has the following syntax:

```
<ARCO_AFProps afId="…" propName="…" constraints="…" varName="…"/>
```

where:

- `afId` is the database identifier of the ARIF exhibition space,
- `propName` is the type of data to be retrieved (see Table 3-2),
- `constraints` provides additional constraints/rules on the data to be retrieved,
- `varName` is the name of a variable where the result is stored.

In Table 3-2, the list of valid `propName` attribute values is presented with the description of the result produced by the `ARCO_AFProps` command and allowed constraints.

PropName value	Result stored in the variable	Constraints	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 exhibition space	-	Integer
`CHILD_AF_IDS`	A list of database identifiers of child exhibition spaces (subspaces).	-	List
`PARENT_AF_IDS`	A list of database identifiers of all parent exhibition spaces, forming a complete path to the space identified by `afId`. The list is ordered from the parent space id to the root space id.	-	List
`AF_PATH`	A full path to the ARIF exhibition space	-	String
`CHILD_CO_IDS`	A list of database identifiers of Cultural Objects assigned to the ARIF exhibition space identified by	VISIBLE=DOMAIN ORDER=CO_NAME	List

	afId.	ORDER=SEQUENCE_NUMBER	
CHILD_CO_IDS_RECURSIVE	A list of database identifiers of Cultural Objects assigned to the ARIF exhibition space identified by afId and its all subspaces.	-	List

Table 3-2. Properties of ARIF Spaces

Retrieving Data Related to Cultural Objects

To retrieve data related to Cultural Objects the ARCO_COProps command is used. The ARCO_COProps element has the following syntax:

```
<ARCO_COProps coId="…" propName="…" constraints="…" varName="…"/>
```

where:

- coId is a database identifier of the Cultural Object,
- propName is the type of data to be retrieved (see Table 3-3),
- constraints provides additional constraints/rules on the data to be retrieved,
- varName is the name of a variable where the result is stored.

In Table 3-3, the list of valid propName attribute values is presented with description of the result produced by the ARCO_COProps command and allowed constraints.

PropName value	Result stored in the variable	Constraints	Type of result
CHILD_MO_IDS	A list of database identifiers of Media Objects associated with the Cultural Object identified by coId.	VISIBLE= DOMAIN	List
CHILD_MO_IDS: mime_types	A list of database identifiers of Media Objects having a particular mime type, associated with the Cultural Object identified by coId.	VISIBLE= DOMAIN	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 coId is of type ‘Refined’, or NULL otherwise.		Integer
PARENT_AF_IDS	The list of database identifiers of the ARIF spaces to which the Cultural Object identified by coId is assigned.		List
HAS_THUMBNAIL	The Boolean value indicating if the Cultural Object has thumbnail image		Boolean

Table 3-3. Properties of Cultural Objects

Retrieving Data Related to Media Objects

To retrieve data related to Media Objects the `ARCO_MOProps` command is used. The `ARCO_MOProps` element has the following syntax:

```
<ARCO_MOProps moId="…" propName="…" varName="…"/>
```

where:

- `moId` is a database identifier of the Media Object,
- `propName` is the type of data to be retrieved (see Table 3-4),
- `varName` is the name of a variable where the result is stored.

In Table 3-4, the list of valid `propName` attribute values is presented with description of the result produced by the `ARCO_MOProps` command.

PropName value	Result stored in variable	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`	Array
`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 `moId` is assigned.	List
`HAS_THUMBNAIL`	The Boolean value indicating if the Media Object has thumbnail image	Boolean
`CHILD_MO_IDS`	A list of database identifiers of Media Objects contained by a composite Media Object identified by `moId`.	List
`CHILD_MO_IDS:mime_types`	A list of database identifiers of Media Objects having a type from the list of `mime_types`, contained by a composite Media Object identified by `moId`.	List

Table 3-4. Properties of Media Objects

Retrieving Visualization Properties

To retrieve visualization properties of ARIF elements: ARIF exhibition spaces, Cultural Objects, and Media Objects the `ARCO_VisProp` command is used. The `ARCO_VisProp` element has the following syntax:

```
<ARCO_VisProp afId="…" coId="…" moId="…" propName="…" varName="…"/>
```

where:

- `afId` is a database identifier of the ARIF exposition space,
- `coId` is a database identifier of the Cultural Object,
- `moId` is a database identifier of the Media Object,
- `propName` is a name of the object property to be retrieved,
- `varName` is the name of a variable where the result is stored.

To retrieve visualization property of an ARIF exhibition space, the `coId` and `moId` must have `null` values. To retrieve visualization property of a Cultural Object, the `moId` must have a `null` value. Otherwise, the visualization property of the Media Object in context of current ARIF exhibition space and particular Cultural Object is retrieved.

Retrieving AMS data

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

```
<ARCO_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 `ARCO_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 inserting 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.

Template Parameterisation Commands

The following X-VRML commands are used for parameterisation of the visualization templates in the final ARCO Prototype: `ARCO_InputText`, `ARCO_InputCheckbox`, and `ARCO_InputSelect`.

Each parameterisation command has the following syntax:

```
<ARCO_Input ControlName common_attributes specific_attributes/>
```

where the *ControlName* is one of the following: `Text` – denoting text field, `Checkbox` – denoting checkbox, or `Select` – denoting selection control. A set of *common_attributes* is presented in the Table 3-5:

Attribute	Meaning	Possible values
`varName`	The name of the X-VRML variable where the value of the control will be stored	-
`category`	The source of data.	'DIRECT_VALUE' | 'GENERAL_DATA' | 'MEDIA_OBJECT' | 'MEDIA_OBJECT_TYPE' | 'TEMPLATE' | 'TEMPLATE_OBJECT' | 'XSL_TEMPLATE'
`dataType`	Type of data	'color' | 'string' | 'Boolean' | 'float' | 'integer' | 'null'
`restriction`	Definition of possible restrictions of values allowed	
`obligation`	Defines whether the value has to be provided or not	'optional' | 'fixed' | 'required'
`defValue`	Default value of the control or variable	-
`afid`	ARIF exhibition space identifier	integer
`domain`	ARIF presentation domain name	string
`class_`	CSS class definition	string
`style`	CSS style definition	string
`enabled`	Is the control enabled	'true' | 'false'
`tabIndex`	Number in the tab-stop sequence	integer
`onClick`	onClick event action	string
`onDblClick`	onDblClick event action	string
`onKeyDown`	onKeyDown event action	string
`onKeyUp`	onKeyUp event action	string
`onKeyPressed`	onKeyPressed event action	string
`onMouseDown`	onMouseDown event action	string
`onMouseUp`	onMouseUp event action	string
`onMouseMove`	onMouseMove event action	string
`onMouseOver`	onMouseOver event action	string
`onMouseOut`	onMouseOut event action	string
`onFocus`	onFocus event action	string
`onBlur`	onBlur event action	string

onChange	onChange event action	string

Table 3-5. Common attributes of parameterisation commands

In Table 3-6 the specific attributes of parameterisation commands are presented.

Attribute	Meaning	Possible values
ARCO_InputText		
editable	Defines if a user may edit the value or enter data	'true' | 'false'
length	A visual length of the control	integer
maxLength	Maximum length allowed for the entered string	integer
isPassword	Defines whether the entered text should be masked with asterisk signs or not	'true' | 'false'
ARCO_InputSelect		
length	The length of the option list	integer
multiple	Defines whether multiple selections are allowed	'true' | 'false'

Table 3-6. Specific attributes of parameterisation commands

3.3. ARCO X-VRML Implementation

3.3.1. ARIF X-VRML Server

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 or XDE files 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 95.

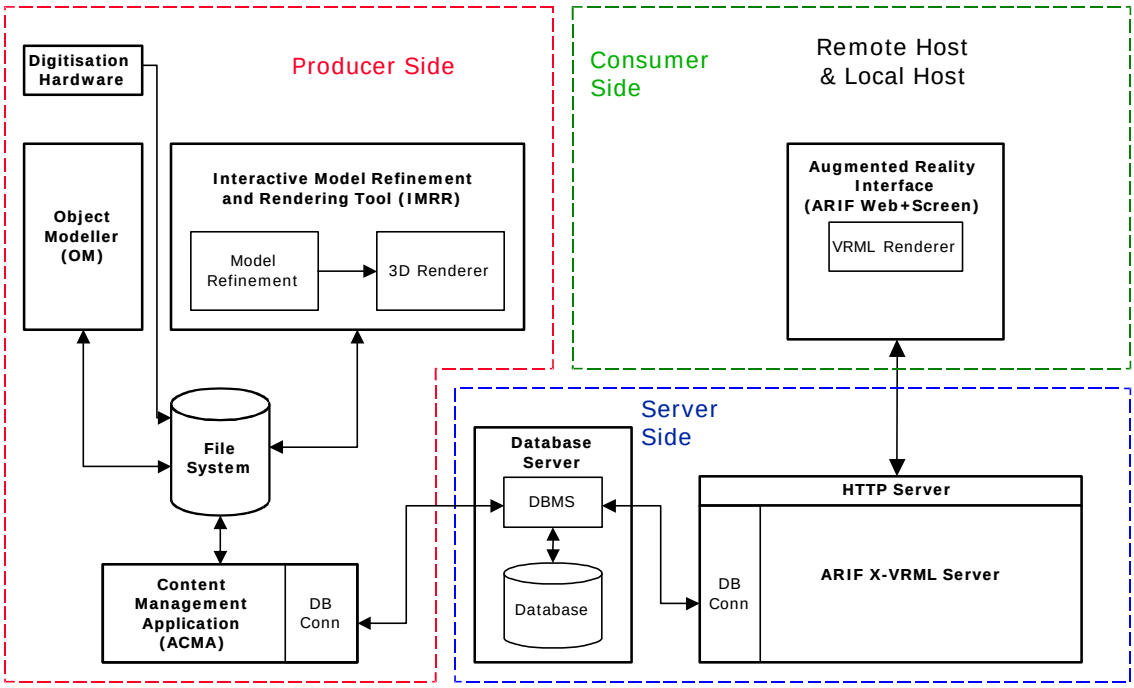


Figure 95. ARIF X-VRML Server in the overall ARCO architecture

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 or XDE files, 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 or XDE files,
- 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 the future 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 or XDE files. 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 or XDE files.

Architecture of the ARIF X-VRML Server

The overall architecture of the ARIF X-VRML Server is presented in Figure 96. The ARIF X-VRML Server consists of the X-VRML Module and the ADAM – ARCO Data Access Module subcomponents communicating through the Data Access Interface with the ARCO Database or XDE Files. These X-VRML Server and ADAM modules are described below.

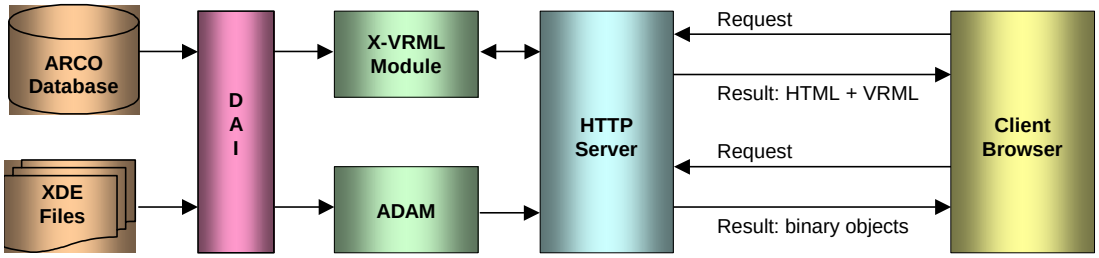


Figure 96. Architecture of the ARIF X-VRML Server

X-VRML Module

Overview of the X-VRML Module

The overall architecture of the X-VRML Module is presented in Figure 97.

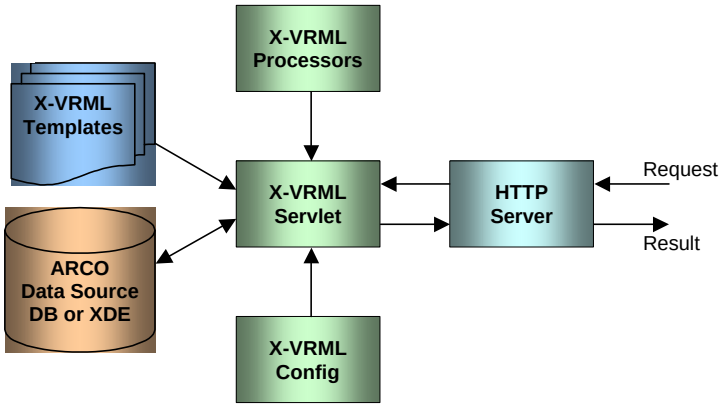


Figure 97. The X-VRML Module

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 or retrieves data from XDE files.

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 [18]. 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.

Internal structure of the X-VRML Servlet

The internal structure of the X-VRML Servlet is presented in Figure 98.

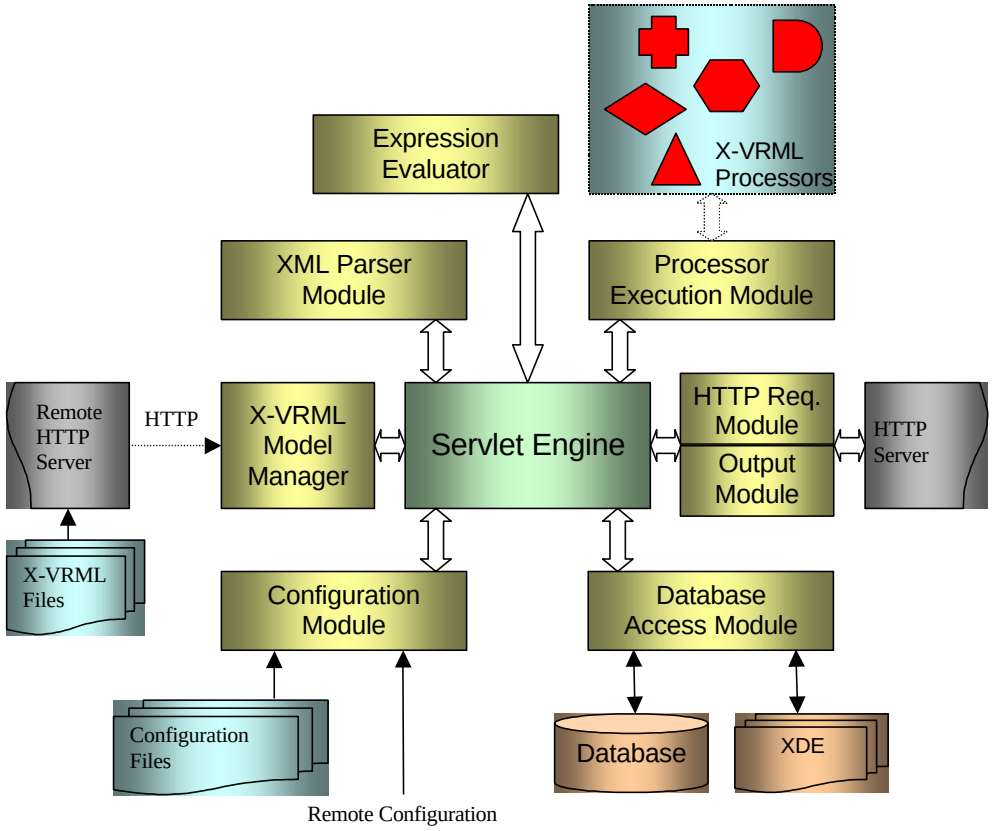


Figure 98. Internal structure of the X-VRML Servlet

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. In the ARCO Third Prototype the database access module will be able to connect to both ARCO Database and XDE files.

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.

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 or XDE files 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 99.

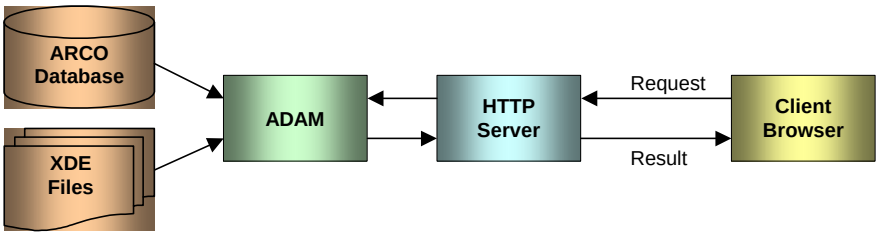


Figure 99. Retrieving binary data by the ADAM subsystem

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 3.2.6).

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 thumbnails of Media Objects is the following:

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

The structure the URL for retrieving thumbnails of Cultural Objects is the following:

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

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

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

where

`http` – is the specification of the communication protocol,

`dns` – is the DNS name of the ARCO server,

`alias` – is the alias to the ADAM servlet (depends on HTTP server configuration),

`nn` – is the identifier of the binary object to be retrieved.

3.3.2. 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 exhibition spaces (ARIF folders). Each ARIF exhibition space may represent an exposition, a part of the exposition related to a particular subject, a museum room, etc. Subspaces may be used to divide exposition into smaller parts, e.g., focused on a particular topic.

The ARIF exhibition spaces consist of ARIF folders containing two types of elements:

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

When an end-user enters an ARIF exhibition all Cultural Objects that are assigned to this particular ARIF folder 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) by the use of a special *Presentation Manager* tool (cf. Section 3.3.3).

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.

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 spaces may be a value of a parameter defining the background image.

In order to speed-up the process of designing ARIF contents and to 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.

To achieve maximum flexibility the concept of ARIF presentation domains was introduced. An ARIF presentation domain is the environment in which the ARIF interface is used. The final ARCO prototype addresses two main web domains: WEB_LOCAL and WEB_REMOTE. The list of ARIF presentation domains is extensible.

Each X-VRML template is associated with a list of ARIF presentation domains. In each ARIF exhibition space multiple instances of templates for different domains may be created (but at most one for one 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. It permits to create different visualizations for different user agent environments, e.g., Web Remote, Web Local, etc.

An example tree of ARIF exhibition spaces displayed in the Presentation Manager is presented in Figure 100. In the '*Victoria & Albert Museum*' ARIF exhibition space, template instances for '*WEB_LOCAL*', '*WEB_LOCAL.INFO*' and '*WEB_REMOTE*' domains are created. These instances are named '*Main Gate*', '*Main Gate Info*' and '*Web Interface*', respectively. The template instance '*Web Interface*' is inherited by all subspaces of the '*Victoria & Albert Museum*' because the subspaces do not override this instance. Such inherited instance is denoted in the Presentation Manager by a grey icon. On the contrary, the '*Web Local*' template has another instance defined in the '*European Art*' space called '*Local Expo*'. Therefore, this space may be visualized differently than the parent space. The '*Local Expo*' instance is valid for all of subspaces in the '*European Art*' space, namely '*Europe & America*', '*Mannerism…*', '*The Medieval Treasury*', and '*Sculpture*', except '*Gothic Art*', which contains its own instance of the '*Web Local*' template called '*Local Room*'. The '*Gothic Art*' space contains also references to cultural objects, which will be visualized when a user enters this exhibition space in one of the ARIF interfaces.

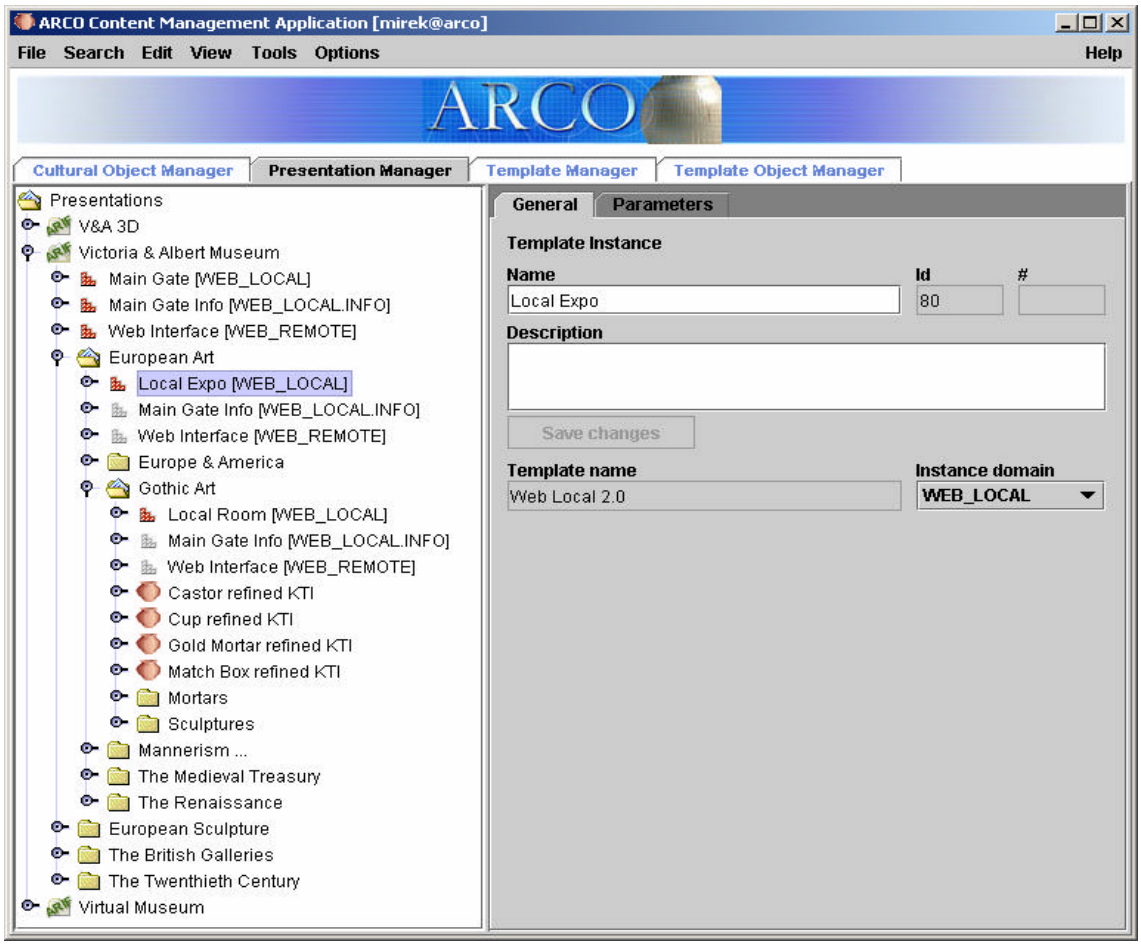


Figure 100. Example of template instances created in the Presentation Manager

Summarizing, the presented 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.

In the following sections the functionality of the Template Manager and the Presentation Manager will be described.

3.3.3. Template Manager

The Template Manager allows administering X-VRML visualization templates stored in the ARCO database. The Template Manager is presented in Figure 101.

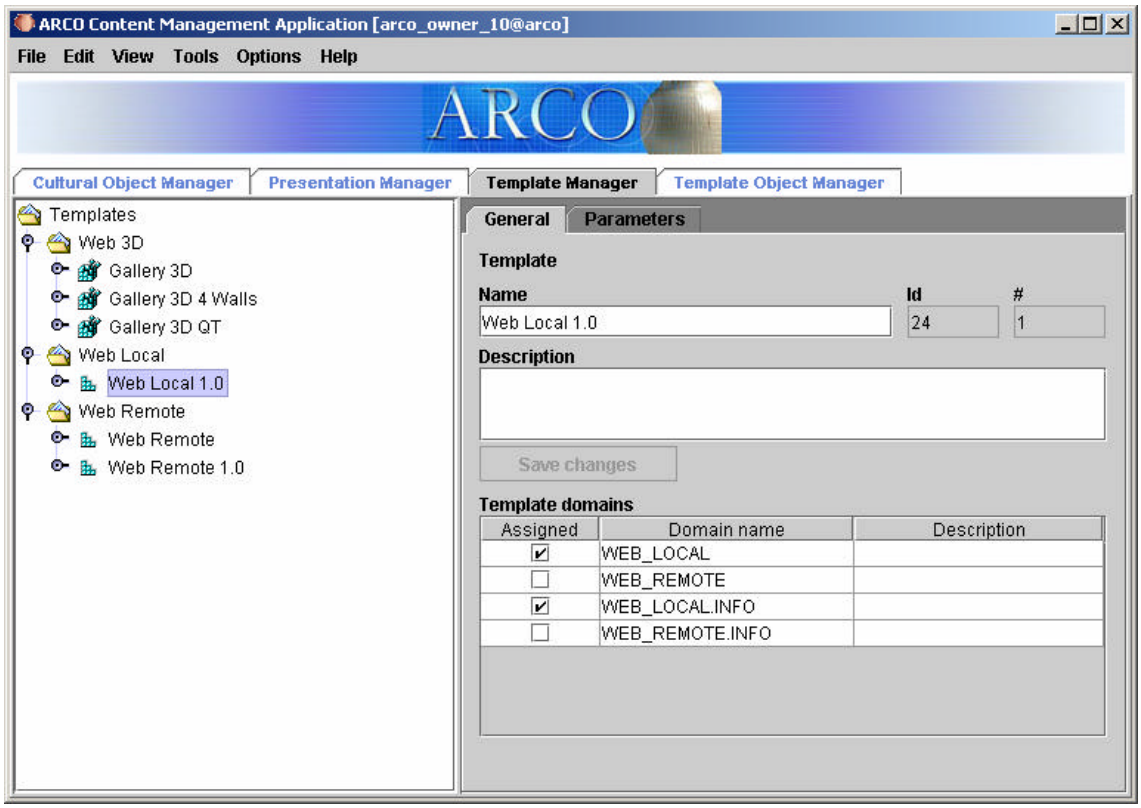


Figure 101.Template Manager within ACMA

The Template Manager is divided into two main panels. The left panel contains a tree that presents structure of 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 can contain other folders and templates. Each template can have number of template parameters. There is an icon next to the name of each template. The icon represents the type of the template (- 2D template, - 3D template).

A detailed information about each selected item in the tree (such as the name, description, database ID, etc.) is displayed in the right panel. For templates these are name and description, database ID and table with presentation domains the template belongs to. In case of a template parameter – the name, label, data type, data type description, and the default value can be accessed. Additionally for templates there is another tab (named 'Parameters') which contains information about parameters, their names, types and values in tabular form. This is the place where default values of 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). An example of a template with a collection of parameters is presented in Figure 102.

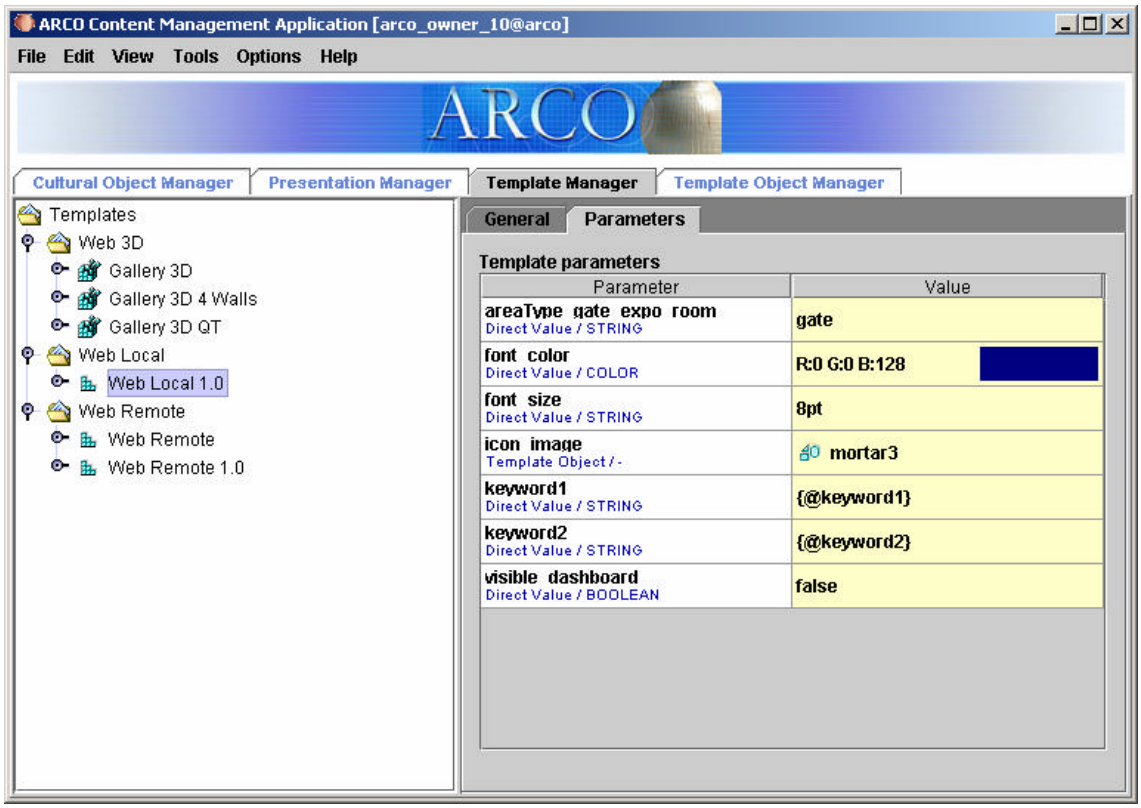


Figure 102.Template with parameters

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

The following actions are accessible for different elements of the tree.

- Folder actions

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

- Template actions

  The name, description, and assignment of a template to presentation domains can be modified. After modifications user can save changes into the database (*Save changes*).

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

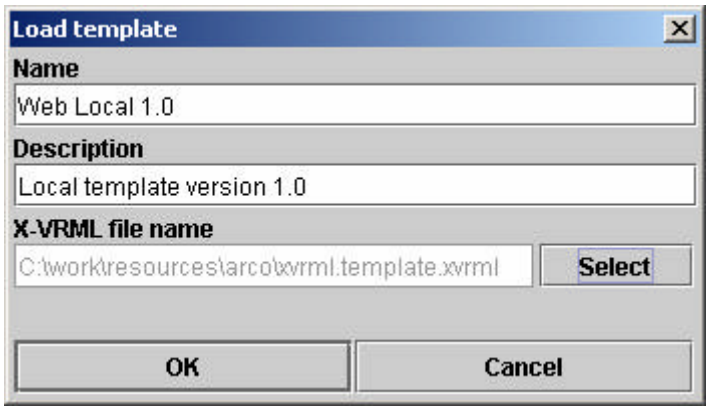


Figure 103.Loading template into database

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

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*). Removing template from the folder fails if the instance of the template being removed is the last one. In such a case the user is prompted to use the delete action instead.

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

Default values of template parameters can be set in the 'Parameters' tab. Double clicking on a value in the parameter table will cause a specialized editor to open.

3.3.4. Presentation Manager

The Presentation Manager allows administering virtual exhibitions stored in the ARCO database. The Presentation Manager is shown in Figure 104.

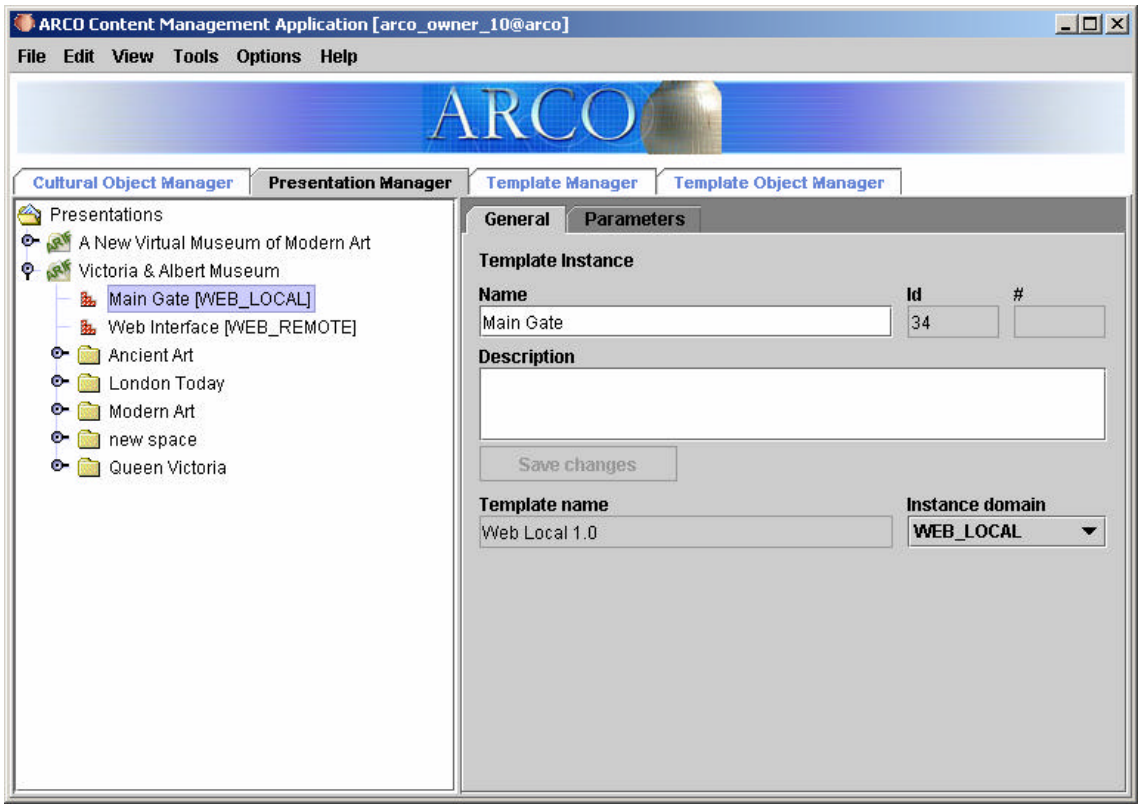


Figure 104.Presentation Manager within ACMA

Virtual exhibitions are defined by a hierarchy of ARIF spaces containing cultural objects and template instances. Multiple template instances corresponding to different domains can be assigned to a single ARIF space.

Each template can have parameters, which can be set in its instances. If values for all template parameters are set the result is a rigid presentation. Otherwise, if not all parameters are provided, the end user will be able (in case of non-required parameters) or will have to (in case of required parameters) to provide the parameter values while the presentation is about to be shown.

The Presentation Manager is divided into two main panels. The left panel contains a tree presenting the structure of ARIF folders (exhibition spaces), 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 a number of embedded template instances. There is an icon next to the name of each object. The icon represents the type of the object (- ARIF presentation, - ARIF space, - instance of 2D template, - instance of 3D template, - inherited instance of 2D/3D template).

Detailed information about each selected item in the tree is displayed in the right panel. For ARIF spaces these are name, description and database ID. For template instances – name, description, template name and instance domain. In case of a Cultural Object the same information as in Cultural Object Manager appears.

For template instances, a tab panel named ‘Parameters’ is displayed containing information about parameters, their names, types and values in a tabular form. This is the place where values of template parameters can be set. The value can be either a number or a string or a certain object (e.g. Media Object, Template Object). An example of template instance with parameters is presented in Figure 105.

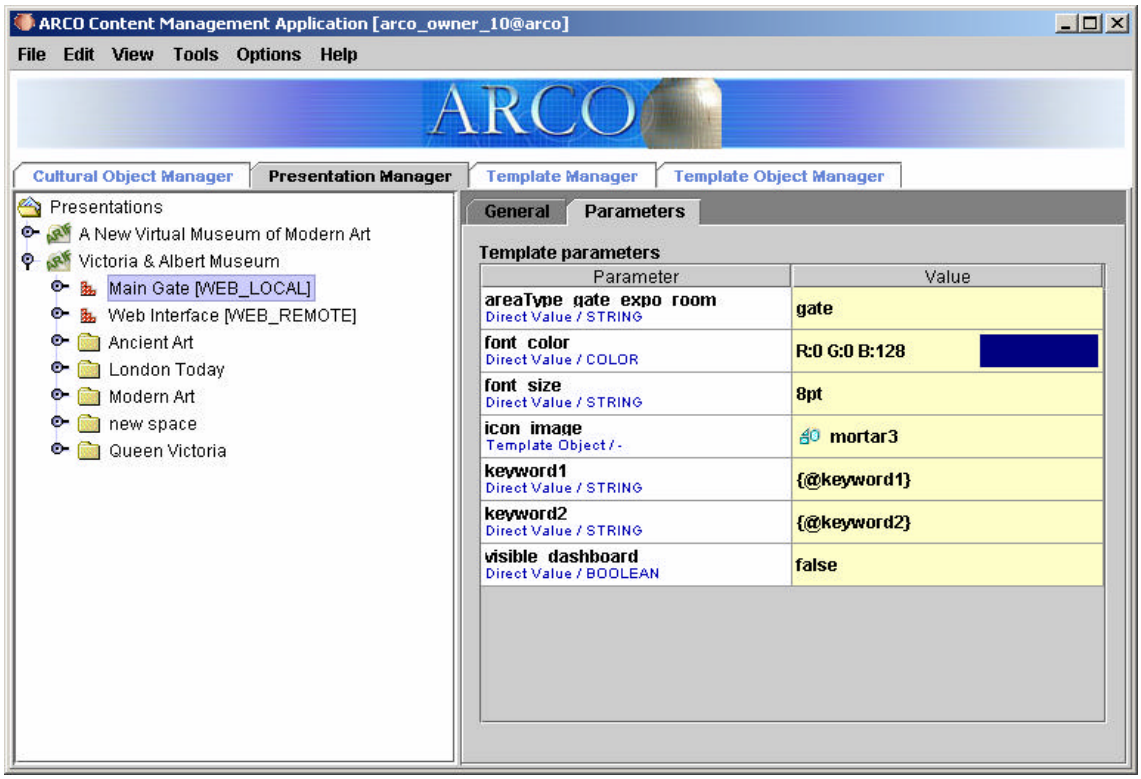


Figure 105. Template Instance and its parameters

For ARIF spaces, Cultural Objects and Media Objects, a tab named ‘Properties’ appears. It allows managing visualization properties of ARIF spaces, Cultural Objects and Media Objects. The value of a property can be a number, a string or a certain object. An example ARIF folder with properties is presented in Figure 106.

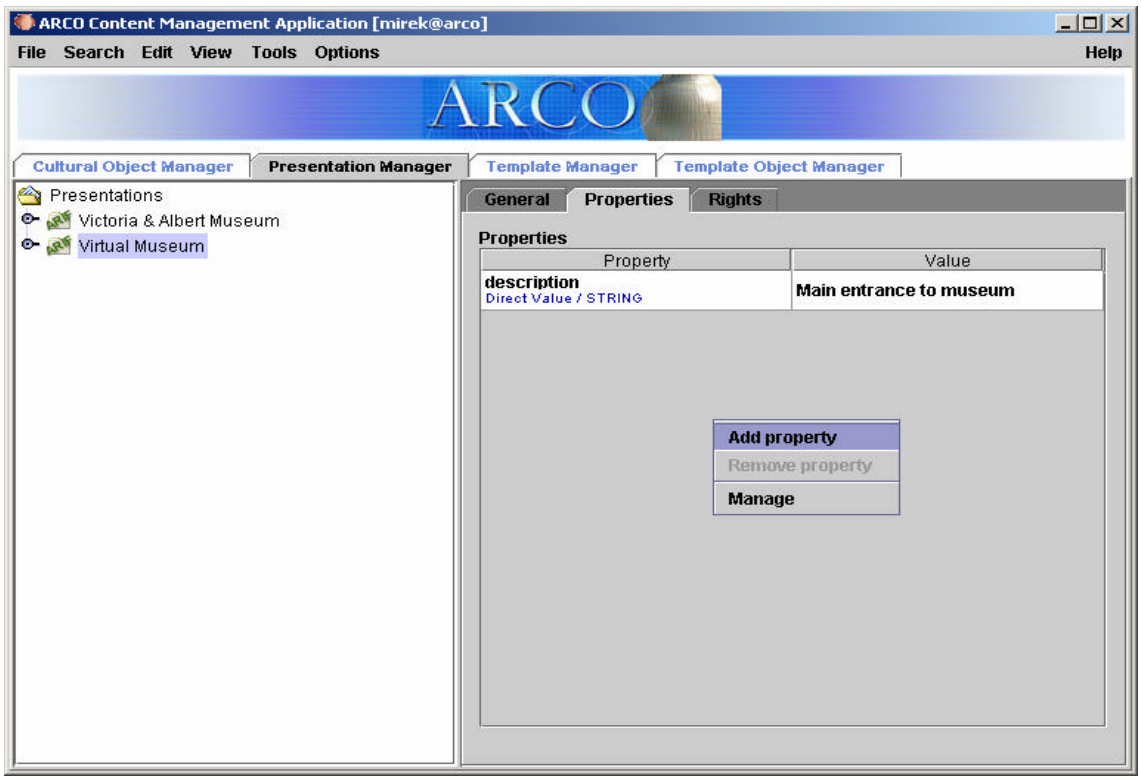


Figure 106. ARIF folder with ‘description’ property set

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

The following actions are accessible for different elements of the ARIF folder tree.

- Folder actions

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

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

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

  Values of properties can be set in the 'Properties' tab. Double clicking on a value in property table will cause a specialized editor to open.

  A user can grant or revoke access rights to an ARIF folder for another user or group of users (*Access rights*).

- Template instance actions

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

  Template instances can be deleted (*Delete*) or shown in a browser (*Show in browser*). When the user wants to show the instance, an Internet browser is launched. The default browser is Internet Explorer. The default setting can be changed in preferences.

  Values of template parameters can be set in the 'Parameters' tab. Double clicking on a value in parameter table will cause a specialized editor to open.

- Cultural Object actions

  Cultural Objects can be removed from presentation space (*Remove*).

  Values of properties can be set in the 'Properties' tab. Double clicking on a value in property table will cause a specialized editor to open.

- Media Object actions

  Values of properties can be set in the 'Properties' tab. Double clicking on a value in property table will cause a specialized editor to open.

The augmented reality interface uses markers for displaying objects [5]. Presentation Manager can be configured to display icons of markers the Cultural Objects are assigned to. The sequence of Cultural Objects can be changed by the *Move up* and *Move down* menu options. In Figure 107, an example presentation with markers is shown.

Figure 107. Presentation Manager with *Show markers* option enabled

3.3.5. Client-side X-VRML Processor

In order to allow museum curators to design virtual exhibitions visually, a 3D authoring interface based on a client side X-VRML processor is used in the final ARCO prototype. The client-side X-VRML processor interprets X-VRML models on the client side, i.e. in a web browser. As opposed to server-side processing, which is performed only once and a fixed VRML model is produced, in the client-side processor, interpretation is performed during the whole life-time of the virtual scene. This allows to exploit the full potential of the X-VRML language – in particular implementation of persistency.

The 3D authoring interface is used to allow museum curators to visually position objects in a 3D exhibition space. Using this interface a museum curator can arrange the objects by simply moving them in the exhibition area. These changes are automatically recorded in the database. When a visitor opens the gallery in the end-user interface the objects will be located exactly as arranged by the museum curator.

In Figure 108, the 'VAM Art Deco Corridor' template displayed in the 3D authoring interface based on the client-side X-VRML processor is presented. The same exhibition area displayed in the end-user interface accessible by museum visitors is presented in Figure 109.

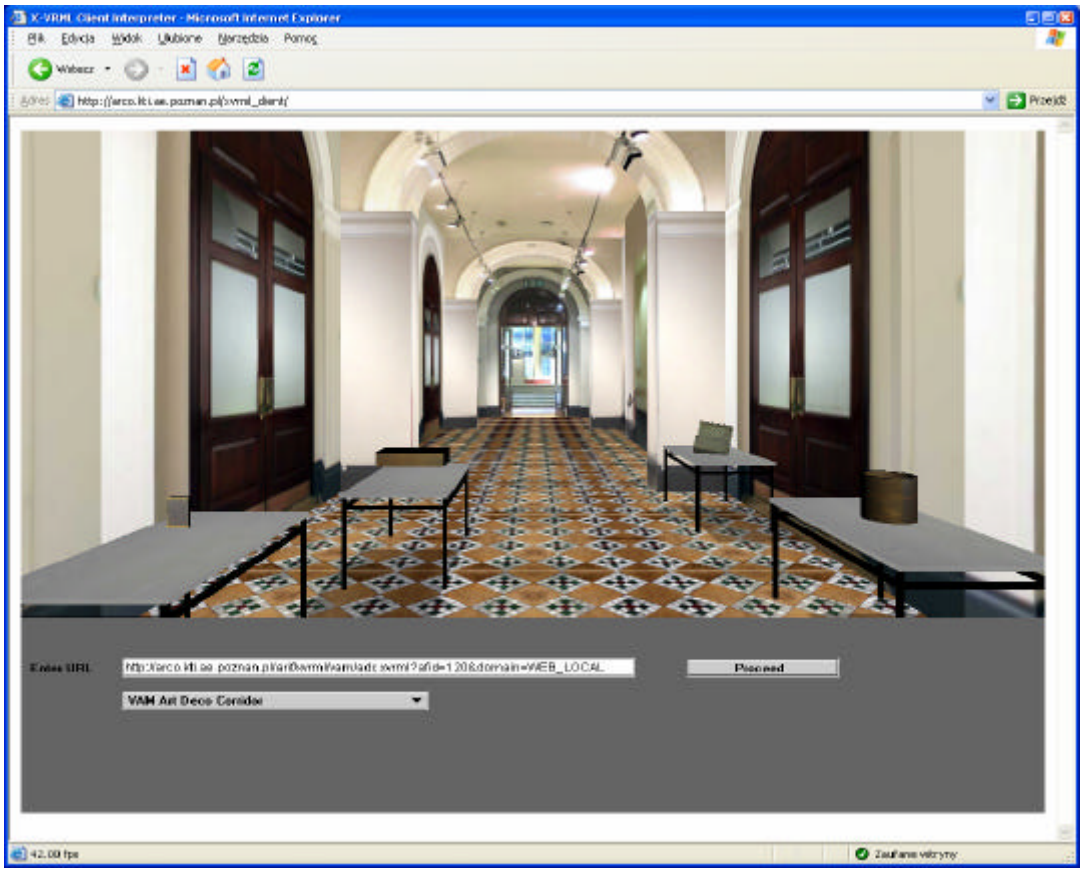


Figure 108. VAM Art Deco Corridor – 3D authoring interface

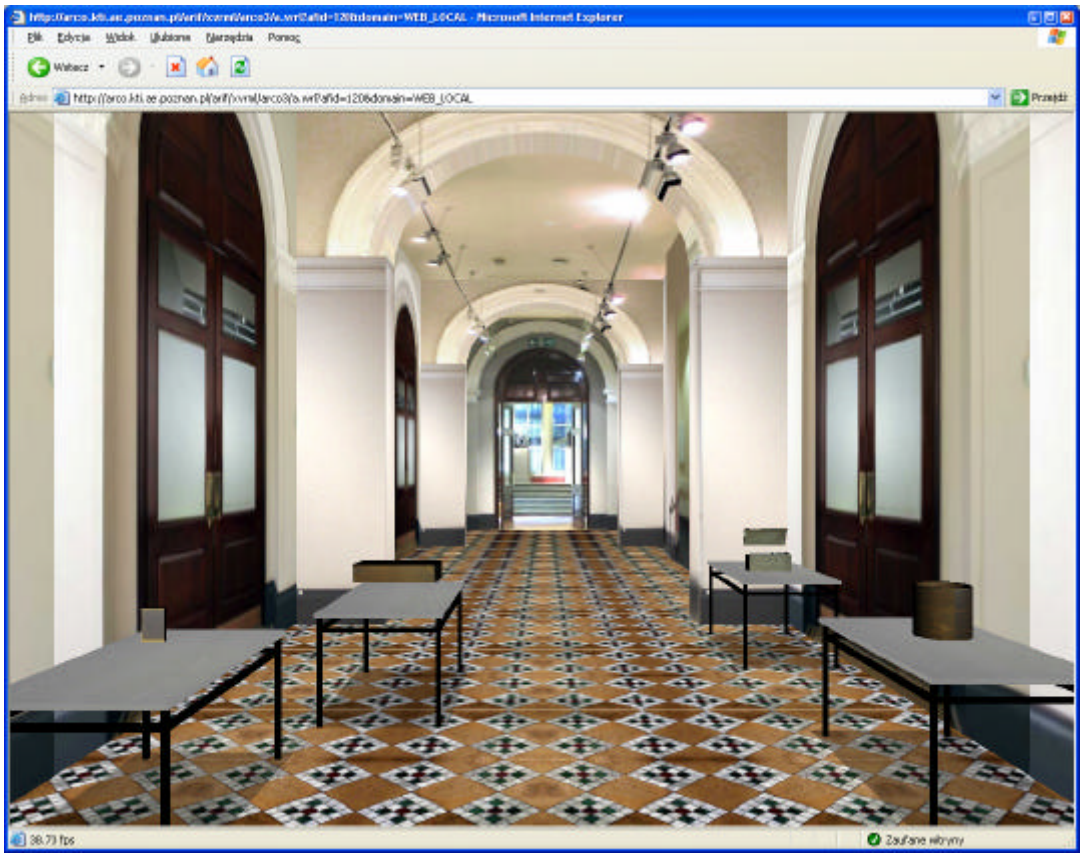


Figure 109. VAM Art Deco Corridor – end-user interface

3.4. XSL Stylesheets in ARCO

3.4.1. Overview

The XSL stylesheets are used within the ARCO system for displaying XML data stored in the ARCO database in a human readable way. In the final 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 the "Report on the XML descriptions of the database of cultural objects " [1].

The XSL stylesheets are used within the ARCO system in connection with the X-VRML templates, which are the base implementation technology for all ARIF Interfaces – *Web Local*, *Web Remote*, and *Augmented Reality Interface* [5]. The X-VRML language provides commands for retrieving XML data from the database and applying XSL stylesheets – also retrieved from the database (cf. Section 3.2.6).

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

3.4.2. X-VRML – XSL Architecture

In Figure 110, the processing of AMS XML data with X-VRML templates and XSL stylesheets in the final ARCO prototype is presented.

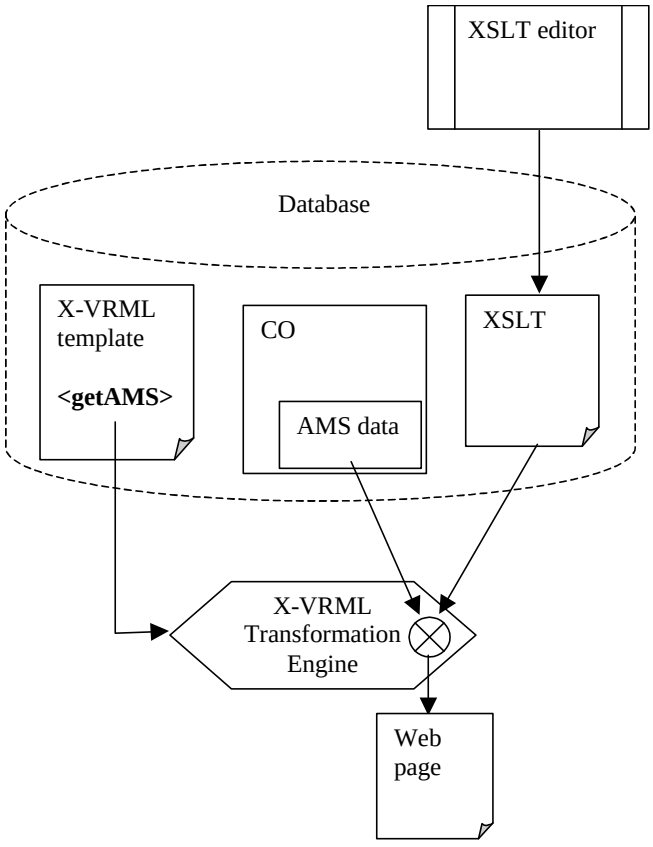


Figure 110. Processing of AMS with X-VRML and XSLT

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 – `ARCO_getAMS` responsible for processing of AMS with XSL (cf. Section 3.2.6). While interpreting the `ARCO_getAMS` command, the X-VRML processor retrieves the AMS data for the requested object (Cultural Object or Media Object) from the ARCO database or an XDE file. Using the *XPath* expression to extract the AMS element of interest, the X-VRML processor applies an XSL stylesheet – also retrieved from the database or XDE. 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 stylesheet to be used and the *XPath* expression may be parameters of the X-VRML template providing maximum flexibility in displaying AMS in ARIF interfaces.

3.4.3. Designing XSL Stylesheets for AMS Metadata

Existence of advanced graphical tools allows users without XSL programming experience to design ARCO XSL stylesheets. The process of creating an XSL stylesheet for ARCO AMS metadata in *Altova Stylesheet Designer* is presented in Figure 111. 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 AMS XML Schema. As a result, an XSLT stylesheet with appropriate element selection and formatting is generated.

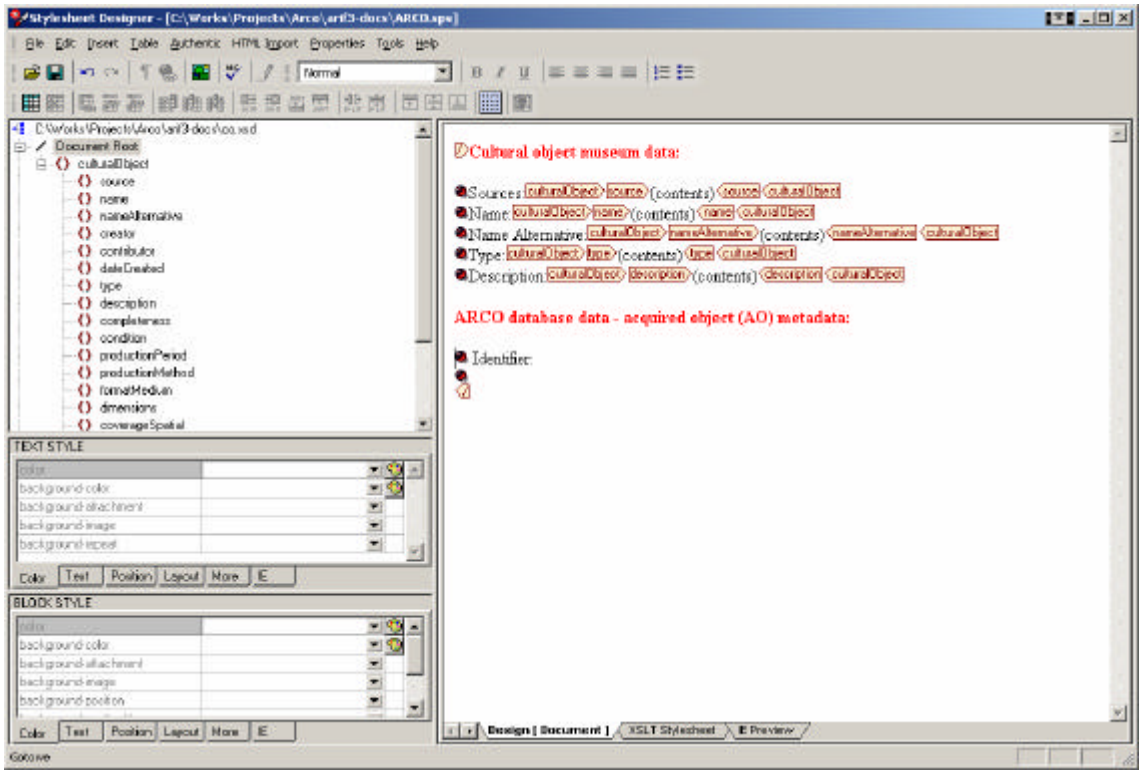


Figure 111. Designing AMS XSL stylesheet in Altova Stylesheet Designer

An XSL stylesheet prepared in such a way may be stored in the ARCO database by the use of the XSL Manager available in ACMA (see Figure 112).

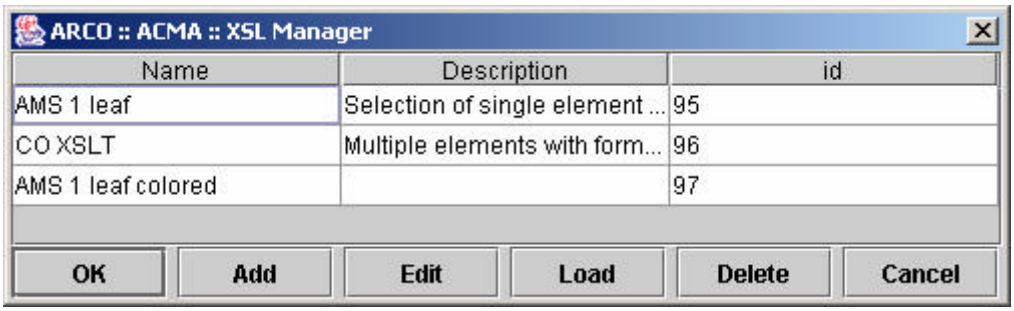


Figure 112. ACMA XSL Manager

Assignment of an XSLT stylesheet to a particular ARIF template may be fixed or parameterised. In the first case a reference to the XSLT stylesheet is included directly within the X-VRML template code. In the second case, the XSLT stylesheet is an X-VRML template parameter and can be selected using either the ACMA Template Manager – the default value, or ACMA Presentation Manger – the parameter value in a template instance (see Figure 113).

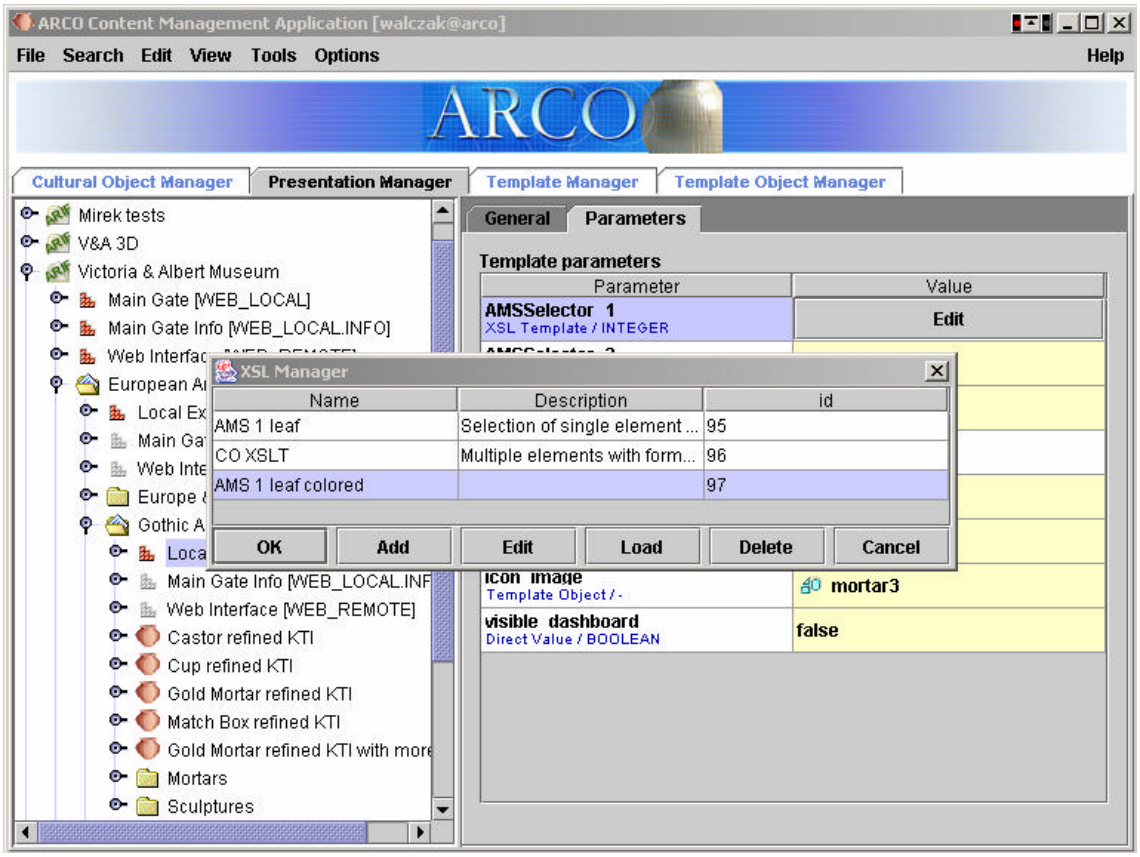


Figure 113. Selecting an XSLT stylesheet as a value of an X-VRML template parameter

The results of processing Cultural Object AMS metadata in the Web Local X-VRML template with different XSL stylesheets, selected in the Presentation Manager, are presented in Figure 114. The results of processing the same Cultural Object AMS metadata with a different XSL stylesheet in the Web Remote X-VRML template is presented in Figure 115.

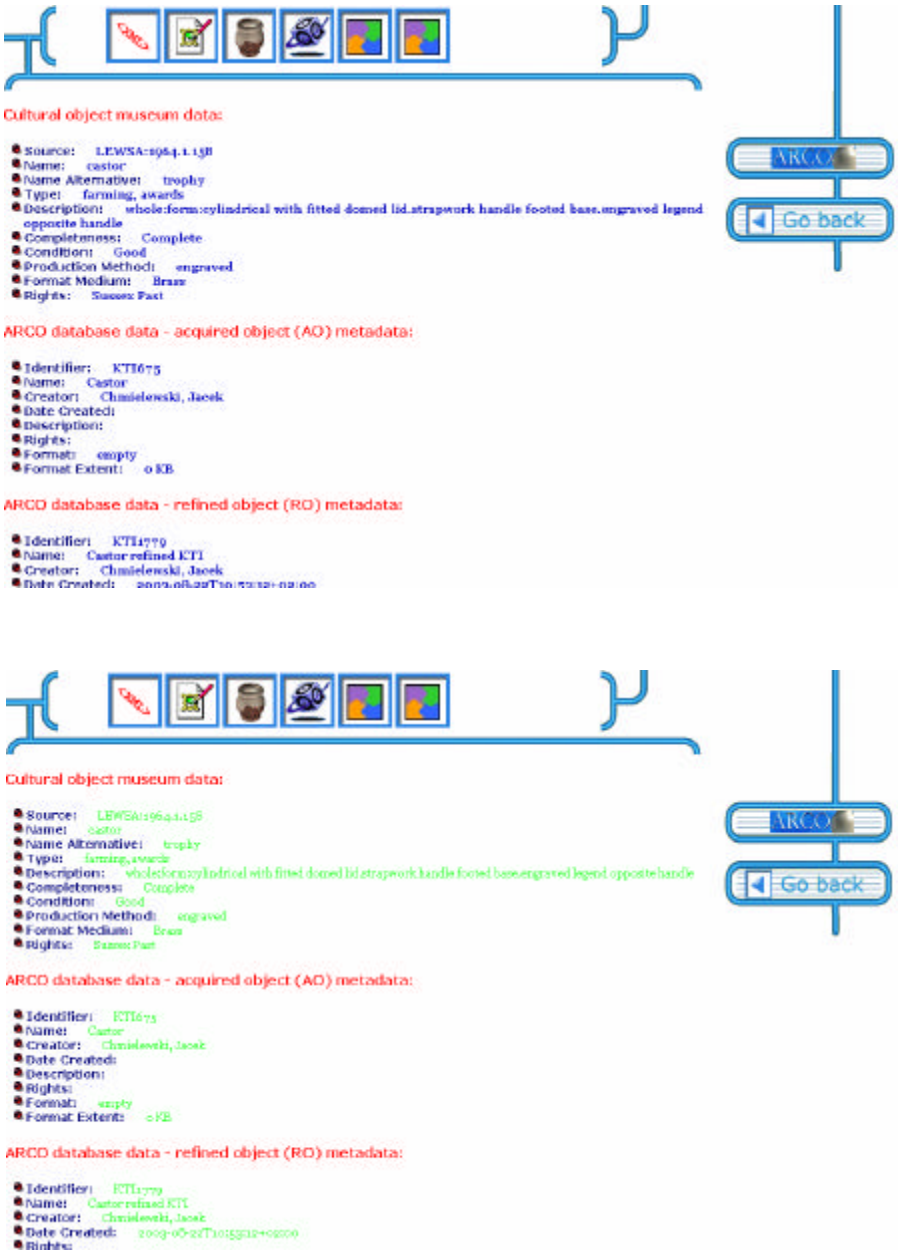


Figure 114. Cultural Object AMS metadata formatted with different XSLT stylesheets in the Web Local X-VRML template

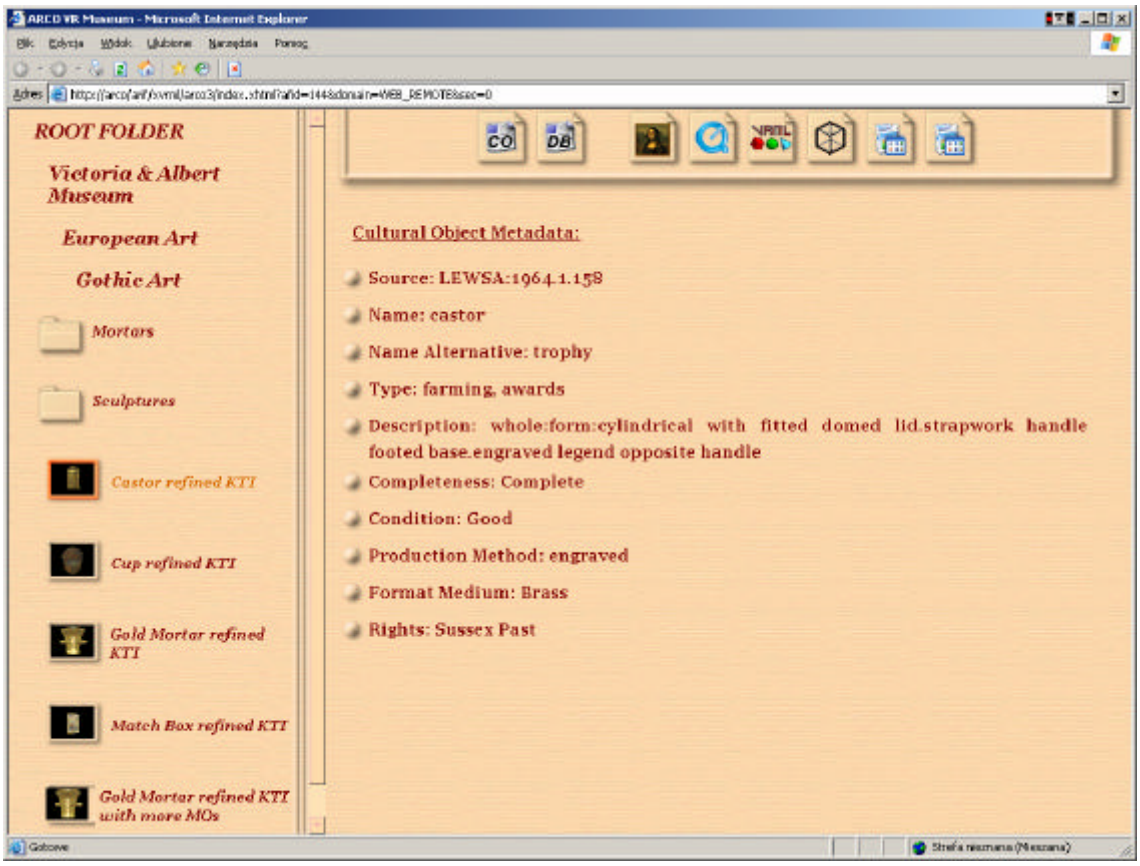


Figure 115. Cultural Object AMS metadata formatted with an XSLT stylesheet in the Web Remote X-VRML template

4. XDE – XML Data Exchange Format

4.1. The concept of XDE

One of the key elements of the ARCO project is the use of XML technologies to enable data interoperability between the components of the ARCO system and between the ARCO system and external systems and applications. Extensive use of XML [17] enables close integration of all ARCO tools into a coherent suite, at the same time providing communication mechanisms that make the ARCO system both internally and externally open.

The use of XML as a communication medium makes the ARCO system internally open. This means that it is possible to replace any of the system components at any stage of the project (or after the project) with a new version or even a new tool as long as the same XML interface is provided. The system will be adaptable to changing user requirements and environments, and it will be possible to gradually upgrade the system to use the state of the art software. Also, it will be possible to include new tools into the ARCO data processing chain. Examples of such tools – that may be of interest for the museums – are image processing, 3D object modelling, model refinement, rendering, and XML editing software.

Use of XML technologies makes the ARCO system also externally open. It means that communication between the ARCO system and other systems and application is possible. This feature is of critical importance for the museum users. After the ARCO system is deployed in pilot sites the museums will start filling the system with the actual data. It is expected that, despite all possible automation in the system, a considerable effort will be required to create a large database of cultural objects. The fact that the ARCO system allows exporting the data to a format understandable to other systems (XML) will ensure that the effort is used in the best possible way.

Eleven different interfaces between ARCO system components have been identified in the final (fourth) ARCO prototype. In order to enable uniform communication on these interfaces, the ARCO system uses a special file format called *XDE – ARCO XML Data Exchange Format*. This file format enables exchange of data between ARCO system components and also between the ARCO system components and the external systems and applications. The XDE data format can carry all kinds of data that is produced and used by ARCO tools.

The XDE Data Exchange Format is based on XML and its structure is specified as XML Schema. The structure of XDE is influenced by the structure of the ARCO database because it is used to carry the same type of data. However, it contains also specific elements that are used only in XDE, and the overall file organization is optimised rather for data exchange than for data storage.

In this section, the description of all interfaces between ARCO system components is provided. Then, the implementation rules that were used for building the XDE XML Schema are discussed. Then a mechanism of mapping XML Schema to database schema is described. Finally, a detailed description of the XDE Data Exchange Format is provided.

4.2. ARCO System Interfaces

4.2.1. Overview of ARCO System Interfaces

The overall architecture of the ARCO system together with all interfaces between system components is presented in Figure 116.

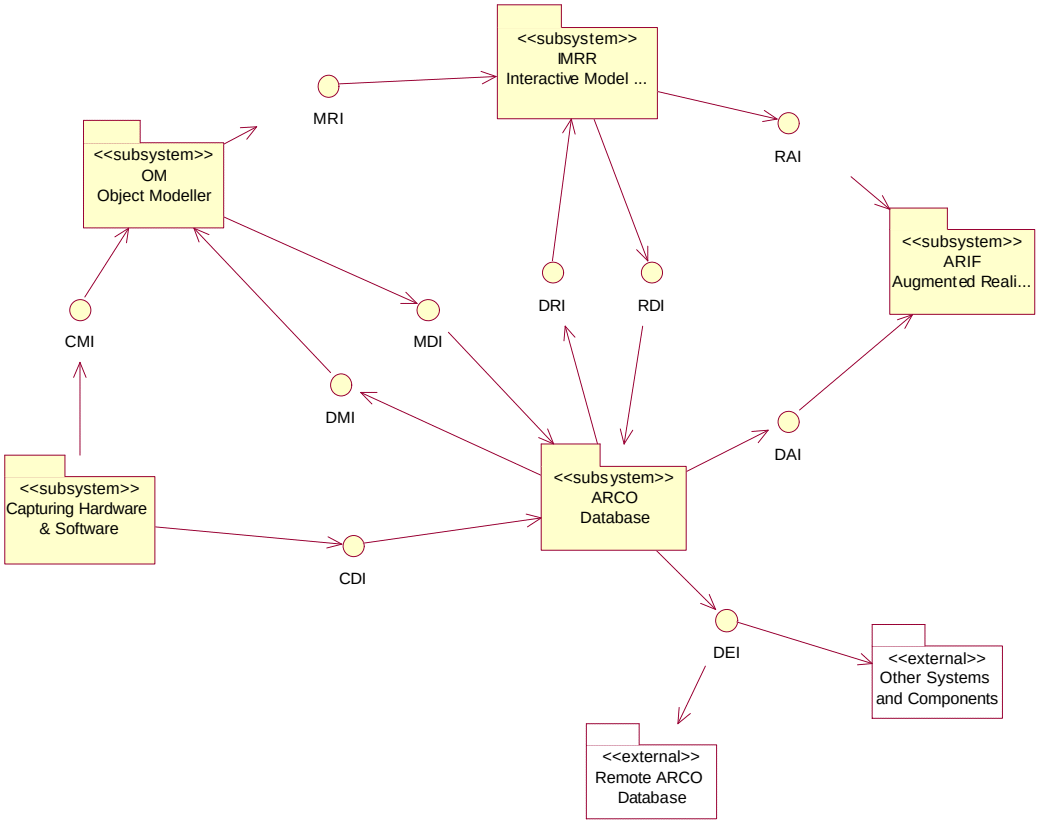


Figure 116. Interfaces of the ARCO system

The ARCO system consists of five main subsystems:

- Capturing Hardware and Software,
- Object Modeller,
- Interactive Model Refinement and Rendering Tool,
- Augmented Reality Interfaces, and
- ARCO Database.

All ARCO subsystems may exchange data with one another and in some cases also with other external systems. Together ten interfaces have been identified in the final ARCO prototype: CDI, CMI, DMI, MDI, MRI, DRI, RDI, RAI, DAI, and DEI as shown in Figure 116. There is also one more interface – GDI (not included in the figure) that has been distinguished as a source of common data used by all tools. All eleven interfaces are shortly described below.

This document describes all possible interfaces between ARCO system components for the purpose of analysis of the required data structures needed to carry the data exchanged between these components. The fact that an interface is described here does not mean that it will be implemented in the ARCO project. During the project lifetime, it will be assessed which interfaces are required and should be implemented.

CDI – Capturing Hardware and Software to Database

The CDI interface is used to transfer data from the capturing hardware and software to the ARCO database. The data includes pictures of the artefact taken during photographic sessions and associated metadata and optionally the cultural object metadata.

CMI – Capturing Hardware and Software to Object Modeller

The CMI interface is used to transfer data from the capturing hardware and software to the Object Modeller. The data includes digital photographs of the artefact and associated metadata.

DMI – Database to Object Modeller

The DMI interface is used to transfer data from the ARCO database to the Object Modeller. The data includes digital photographs of the artefact and associated metadata.

MDI – Object Modeller to Database

The MDI interface is used to transfer data from the Object Modeller to the ARCO database. The data includes the 3D model of the artefact (together with textures), 3D model metadata, Object Modeller project file, and optionally description and metadata for the cultural object.

MRI – Object Modeller to Interactive Refinement and Rendering Tool

The MRI interface is used to transfer data from the Object Modeller to the Interactive Refinement and Rendering Tool – IMRR. The data includes the 3D model of the artefact together with textures and 3D model metadata.

DRI – Database to Interactive Refinement and Rendering Tool

The DRI interface is used to transfer data from the ARCO database to the Interactive Refinement and Rendering Tool. The data includes the 3D model of the artefact, 3D model metadata, all Media Objects associated with the Cultural Object (images, descriptions, models) with their metadata and optionally cultural object metadata.

RDI – Interactive Refinement and Rendering Tool to Database

The RDI interface is used to transfer data from the Interactive Refinement and Rendering Tool to the ARCO database. The data includes the IMRR project file with associated metadata, 3D model of the artefact with textures and associated metadata, cultural object metadata, and optionally description of artefact and/or 3D model.

RAI – Interactive Refinement and Rendering Tool to Augmented Reality Interface

The RAI interface is used to transfer data from the Interactive Refinement and Rendering Tool to the Augmented Reality Interface. The data includes the 3D model of the artefact with associated metadata, cultural object metadata, and optionally description of the artefact and/or 3D model.

DAI – Database to Augmented Reality Interface

The DAI interface is used to transfer data from the ARCO database to the Augmented Reality Interface. The interface carries all contents of the ARCO database required by ARIF Server for the dynamic creation of ARIF content. These include information about folders, Cultural Objects, Media Objects, and associated metadata.

DEI – Data Exchange Interface

The Data Exchange Interface (DEI) is used to exchange data between instances of ARCO system, and also between ARCO system and other systems applications. This includes all ARCO data.

GDI – General Data Interface

The GDI interface is used by ARCO tools when they are working off-line (i.e., without database connection). The interface carries all static ARCO data (data that is rarely changing). These include Cultural Object AMS Schemas, Media Object AMS Schemas, Media Object Types, Associations, Association Parameters, MIME Types, Template Objects Types, Template Objects Parameters, and configuration data. These data are not produced by any ARCO

subsystem but are loaded into the ARCO database by means of the ARCO Content Management Application.

In order to enable the ARCO tools to work off-line – in case of a lack of on-line database connection – these data must be taken from a file.

4.2.2. Capturing Hardware and Software

The capturing hardware and software is used to create digital photographs (images) of artefacts. The images can be described with accompanying metadata – both curatorial and technical [1]. The images and metadata can be transferred to the ARCO database using the *CDI* interface [4].

The digital images created by the Capturing Hardware and Software can be also transferred directly to the Object Modeller. For this purpose, the *CMI* interface is used.

4.2.3. Object Modeller

The Object Modeller is used to construct 3D representations of artefacts from sets of images [2]. Images may be loaded from the ARCO database using the *DMI* Interface or directly from the Capturing Hardware and Software with the *CMI* interface.

The resulting 3D representation of the artefact may be a VRML representation of the object or a 3D Studio MAX model or, as well, both. The VRML representation is composed of a VRML model and a set of texture images. The 3D Studio MAX project is composed of the 3D model scene and a set of texture images.

As a result, the Object Modeller produces a set of complex Media Objects for 3D models: 3D Studio MAX project composite Media Object and VRML Model composite Media Object.

The 3D Studio MAX project Media Object contains .max project file. Its sub-Media Objects of type Simple Image contain required materials images, as shown on Figure 117. The association between 3DS project MO and its sub-MOs, has an attribute whose value is set to the relative path between 3DS materials directory and original location of images.

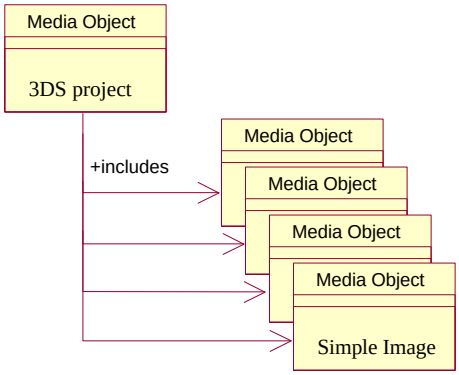


Figure 117. 3D Studio MAX Composite MO and its sub Media Objects

The composite Media Object of VRML model contains the VRML file. Its sub-Media Objects of Simple Image type contain required texture images, as shown on Figure 118. The association between VRML model MO and its sub MOs contains the attribute which value is set to the relative path to texture images for the VRML file.

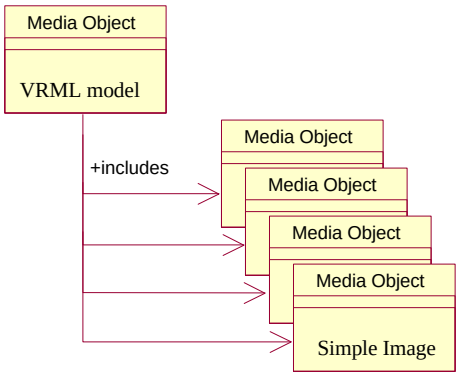


Figure 118.VRML Model Composite MO and its sub Media Objects

The 3D representation of the artefact produced by the Object Modeller can be transfer to the ARCO database for storage by means of the *MDI* interface.

The 3D representation may be also transferred directly from the Object Modeller to the Interactive Refinement and Rendering Tool by means of the *MRI* interface.

4.2.4. Interactive Model Refinement and Rendering Tool

The Interactive Model Refinement and Rendering Tool – IMRR [3] is used to refine and to render 3D representations of Cultural Objects originally created by the OM tool. The original 3D representation is typically stored inside the ARCO database and can be transferred to the IMRR tool by means of the *DRI* interface. In some cases, the 3D representation can be also transferred directly form the Object Modeller tool to the IMRR tool by the use of the *MRI* interface.

The resulting augmented representation is then typically stored in the ARCO database. The data transfer from the IMRR tool to the database may be performed by means of the *RDI* interface.

The IMRR tool can also transfer the 3D representation of the artefact together with metadata information directly to the Augmented Reality Interface – ARIF Server. The data transfer may be accomplished using the *RAI* interface.

4.2.5. Augmented Reality Interface – ARIF

The Augmented Reality Interface [5] is used to visualise the digital representations of cultural objects to the end users. The content to be displayed in ARIF is dynamically created by the ARIF X-VRML Server (XTE). The ARIF Server will be in most cases directly connected to the ARCO Database. In some cases, however, it may be required that the ARIF Server works without direct connection to the ARCO database. This feature could be used, e.g. to demonstrate a virtual exposition from a notebook computer without a database connection.

To enable operation of the ARIF Server without access to the database, it must be provided with a wide range of data either from the ARCO database by the use of the *DAI* interface or the IMRR tool by the use of the *RAI* interface.

4.2.6. ARCO Database

ARCO Database is a common repository for all kinds of data used in the ARCO system [4]. The ARCO Database serves as a data source for the *DMI*, *DRI*, and *DAI* interfaces.

There can be more then one ARCO database instance (e.g., ARCO systems installed in different museum pilot sites). The possibility of exchanging data between ARCO instances is of critical importance for the usability of the system. Also, possibility of exchanging data between ARCO systems and external systems and applications is very important.

To enable exchange of data between the ARCO system and the external world – other ARCO instances or other systems – the *DEI* interface is provided. The *DEI* interface enables the whole or partial (projection and selection) database contents to be exported and imported.

4.3. The Concept of XML Data Exchange Format

In the previous section, eleven different interfaces between ARCO system components have been described. In order to avoid implementation of multiple different data formats for use over these interfaces, at the same time still maintaining the possibility for system components to communicate with each other, the ARCO system uses a common file format called *XDE – ARCO XML Data Exchange Format*. This file format enables uniform exchange of data between the ARCO system components and also between the ARCO system components and the external systems and applications. The XDE data format may carry all kinds of data that are produced and used by particular ARCO tools.

The exchange of data between system components is presented in Figure 119. The abstract XDE file is the central element within this architecture. All components can read or write data structured according to the XDE specification. For example, the Object Modeller can store its results into the XDE file. The file can be then either imported into the ARCO database, or used by the IMRR tool as the input. As a result, the IMRR tool can produce a new version of the XDE file. The resulting XDE file may be either imported into the database or used directly by the ARIF interface for displaying.

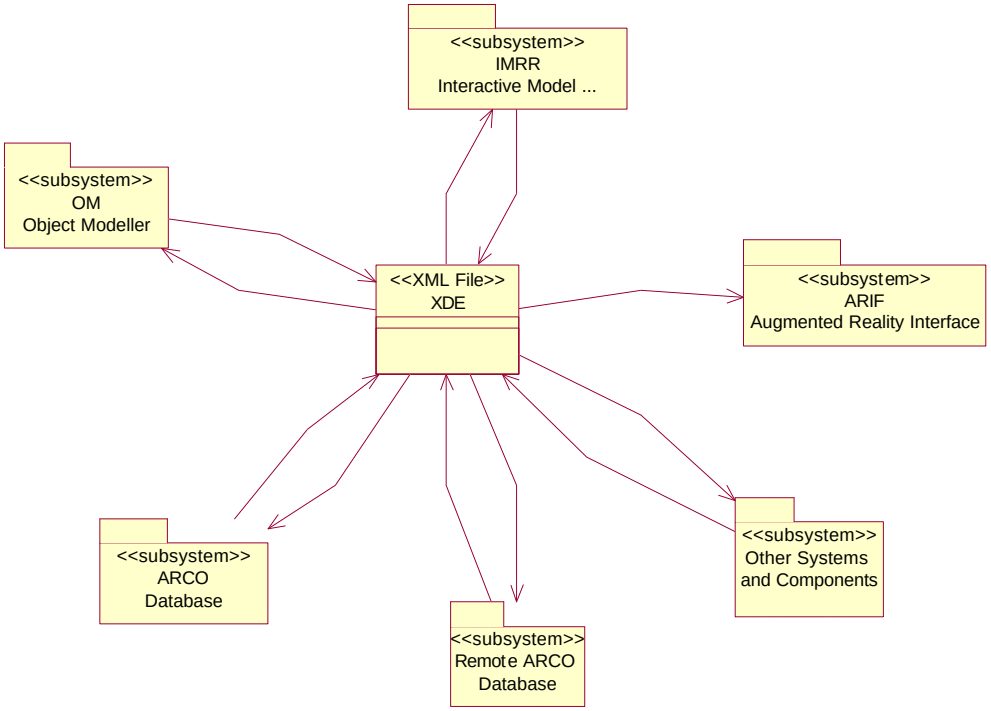


Figure 119.ARCO Data Exchange based on XDE

Since both the ARCO Database and the XDE Data Exchange Format serve as a communication and storage medium for different ARCO components the structure of the XDE file is strongly influenced by the structure of the ARCO database. In fact, the XDE file content is a projection and selection of data coming from the ARCO database or data with similar structure produced by other ARCO components. In addition to the data structures mapped from the ARCO database, the XDE file format contains also specific information used only in XDE such as source, creation date, author, primary contents, etc.

4.4. XDE Implementation Rules

In order to maintain completeness of the description, all parts of the ARCO database have been modelled within the XDE XML Schema. However, various parts of the database are modelled in different ways to optimise use of XDE as a data exchange medium in the ARCO system.

In the process of modelling ARCO database in the XDE XML Schema the following rules were applied:

- All data types used by XDE elements and attributes (including types that correspond to built-in primitive types) are defined in the XDE schema. This simplifies the maintenance of all data types used in XDE.
- Each table in the ARCO database has a corresponding element in the XDE XML Schema;
- Table columns are modelled as attributes, with following exceptions:
    - XML data columns are modelled as child elements;
    - Binary data columns are modelled as child elements;
    - Columns that are mutually exclusive (only one can be not-null) are modelled as child elements of a choice element, so that only one child can be instantiated. This approach has been used for: `MO`, `APV`, `ASSOCP`, `PV`, `TPV`, and `TOPV` elements;
    - Most frequently used foreign key references are modelled by nested elements in XML. Elements used for modelling columns being originally foreign keys, have attributes with the same name as the corresponding foreign key in the ARCO database. The element name has a different name than the foreign key.
- All database table primary keys and foreign keys are modelled as attributes with type `xde:AT_ID`;
- All table elements, to which access based on a foreign key is required, have a primary key constraint with the same name as primary key in the corresponding database table. The constraint is modelled as `<key>` element in the XML Schema with the ’field’ element selecting an attribute with the same name;
- All attributes that model database foreign keys have a foreign key constraints with the same name as the foreign key in the corresponding database table. The constraint is modelled as a `<keyref>` element in the XML Schema with the ‘field’ element selecting the appropriate attribute;
- All database attributes that have a constraint on the list of allowable values are modelled in the XML Schema as enumerated list types;
- Database columns of XML_TYPE type that are used to store XML documents in the database are modelled as complex types with ‘`any`’ element;
- The many-to-one relationships between database tables are modelled as optional unbounded child elements on the "one" side.
- If a table is in a many-to-one relationship with more then one table, it is modelled as a child element of one of these table elements (with the most frequently used relationship), and accessible from the other table elements by the use of a foreign key;
- If the database *table3* is a representation of a many-to-many relationship between *table1* and *table2*:

- The *table3* is modelled as a sequence of optional and unbounded child elements in the element modelling *table1*.
- The child element contains an attribute, which is a foreign key to the element modelling *table2*;

This approach has been used in the following relationships:

- CO (A_CULTURAL_OBJECTS) to MO (A_MEDIA_OBJECTS) with CO_MO (A_CULTURAL_OBJ_TO_MEDIA_OBJ);
- MOT (A_MEDIA_OBJECT_TYPES) to MIMETYPE(A_MIME_TYPES) with MOT_MIMETYPE (A_MEDIA_OBJ_TYPES_TO_MIME_TYPE);
- MOT (A_MEDIA_OBJECT_TYPES) to MOT (A_MEDIA_OBJECT_TYPES) with ASSOC (A_ASSOCIATIONS);
- TOT (A_TEMPLATE_OBJECT_TYPES) to MIMETYPE(A_MIME_TYPES) with TOT_MIMETYPE (A_TEMPL_OBJ_TYPES_TO_MIMETYPES);
- COF (A_CULTURAL_OBJ_FOLDERS) to CO (A_CULTURAL_OBJECTS) with CO_COF (A_CULTURAL_OBJ_IN_FOLDER);
- AF (A_ARIF_FOLDERS) to COF (A_CULTURAL_OBJ_FOLDERS) with COF_AF (A_CO_FOLDERS_IN_ARIF_FOLDER)
- AF (A_ARIF_FOLDERS) to CO (A_CULTURAL_OBJECTS) with CO_AF (A_CULTURAL_OBJ_IN_ARIF_FOLDER);
- TF (A_TEMPLATE_FOLDERS) to XVRML_TEMPLATE (A_TEMPLATES) with TEMPLATE_TF (A_TEMPLATES_IN_FOLDER);
- XVRML_TEMPLATE (A_TEMPLATES) to TD (A_TEMPLATE_DOMAINS) with TP_DOMAIN (A_TEMPLATES_IN_DOMAINS);
- TOF (A_TEMPL_OBJ_FOLDERS) to XVRML_TEMPLATE (A_TEMPLATES) with TO_TOF (A_TEMPL_OBJ_IN_FOLDER);
- TO (A_TEMPLATE_OBJECTS) to TOTP (A_TEMPL_OBJ_TYPE_PARAMS) with TOPV (A_TEMPL_OBJ_PARAM_VALUES);
- MO (A_MEDIA_OBJECTS) to ASSOCP (A_ASSOC_PARAMETERS) with APV (A_ASSOC_PARAM_VALUES);

4.5. XDM – XDE Data Mapping

4.5.1. The Concept of XDM

To enable implementation of generic data import and export tools independent of the specific database structure and the XML schema an extension of the XML schema called *XDM – XDE Database Mapping* has been designed. The extension provides information about mapping of XML elements and attributes to database tables, attributes and sequences. Also, default and error actions are specified. ARCO tools relay on this information to perform XDE import/export operations.

The XDM instances are located in the application information elements <appinfo> of the annotation element <annotation> in each XDE element schema definition. In the following sections detailed description of the XDM extensions of the XDE XML schema is provided.

4.5.2. Overview of XDM

The XML schema defines structure of the XDE XML files. This structure is compatible with the database structure, but there is no information in the raw XML Schema on the mapping between the schema elements and the database elements such as tables, attributes and sequences.

To allow maximum flexibility and adaptability of the software in the future versions, the XDE import and export implementation should be independent of the database structure and XDE schema. To enable such a generic implementation, some additional information needs to be included in the XDE XML Schema for the import and export tools. This information is provided by the *XDM – XDE Database Mapping*. The XDM information is stored in the `<appinfo>` elements of the annotation elements in each XDE element schema definition.

For each element, the XDM section provides the following information:

- whether the element is an XDE structural element (e.g., XDE root) or a database related element (e.g., a Media Object);
- mapping of the XML element and its attributes to the database table and attributes (required only if the names in XML and database are different);
- for database foreign keys modelled in XDE as child elements the information about the connecting database attributes;
- the database sequences used to generate unique values for element attributes while importing the data;
- attributes used to identify elements already existing in the target database while importing data (used for updating data);
- the default actions for importing (e.g., update, insert new);
- the conflict actions while importing (e.g., skip, insert new, ask a user).

The overall schema of the XDM elements is provided in Figure 120.

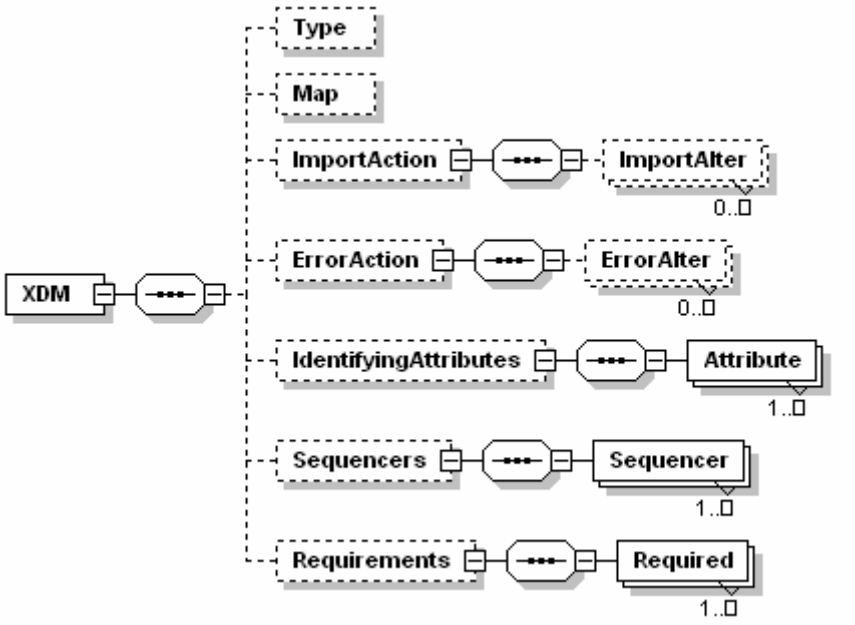


Figure 120.Overall structure of the XDM schema

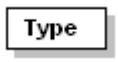
4.5.3. XDM Implementation in XML Schema

XDM Data Types

The following data types are defined in XDM:

- `XDM_ElementTypes` – used to define the XDE element type. The `XDM_ElementTypes` type is a restriction of the `xs:string` type (where `xs="http://www.w3.org/2001/XMLSchema"`). Allowed values of the `XDM_ElementTypes` type are the following:
  - `StructualElement` – the XDE element is a structural element and it is not related to any database value;
  - `ExportableElemnt` – the XDE element represents a table in the database;
  - `ElementAsAttribute` – the XDE element represents an attribute of a database table;
- `XDM_ImportActions` – used for XDM elements specifying default and conflict actions performed during the XDE import. The `XDM_ImportActions` type is a restriction of the `xs:string` type). Allowed values of the `XDM_ImportActions` type are the following:
  - `AddNew` – the imported element will be added as new row in the database using specified *Sequence*;
  - `UseOld` – the imported element will replace an existing row in the database (the row having the same values of the *IdentifyingAttributes* will be replaced);
  - `Skip` – the element will not be imported;
  - `AskUser` – the import procedure will ask a user to choose one of the following actions: `AddNew`, `UseOld`, `Skip`.
- `XDM_TableName` – used for XDM attributes referring to table names in the database. It is defined as an restriction of the `xs:string`.
- `XDM_AttributeName` – used for XDM attributes referring to attribute names in the database. It is defined as an restriction of the `xs:string`.

Element: Type



The `Type` element specifies import and export actions for each XDE element. The XDE element with the `StructualElement` type is not related to the database structure, thus is exported as an fixed XML element and it is ignored during XDE import. The XDE element having `ExportableElement` type is exported using data retrieved from the database and imported to the target database. The XDE element having `ElementAsAttribute` type corresponds to a database attribute, thus it is exported using data retrieved from the database and imported into the target database.

Attributes:

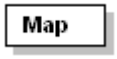
Name	Type	Use
value	XDM_ElementTypes	required

The implementation of the `Type` element is shown in Figure 121.

```
<xs:element name="Type" minOccurs="0">
  <xs:complexType>
    <xs:attribute name="value" type="XDM_ElementTypes" use="required"/>
  </xs:complexType>
</xs:element>
```

Figure 121.Implementation of the `Type` element

Element: Map



The `Map` element defines the name mapping between the XDE elements and attributes and database tables and attributes. If the XDM `Map` element is not defined, the XDE names are used as names for database tables and attributes.

Attributes:

Name	Type	Use
name	xs: XDM_AttributeName	required

The implementation of the `Map` element is shown in Figure 122.

```
<xs:element name="Map" minOccurs="0">
  <xs:complexType>
    <xs:attribute name="name" type="XDM_AttributeName" use="required"/>
  </xs:complexType>
</xs:element>
```

Figure 122.Implementation of the `Map` element

Element: IdentifyingAttributes

The `IdentifyingAttributes` element defines the set of database attributes used during import for detecting that the object already exists in the target database. Each child `Attribute` element defines one attribute name.

Child element: `Attribute`

Attributes:

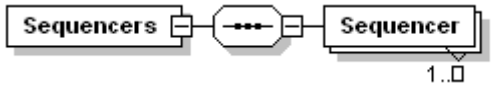
Name	Type	Use
name	XDM_AttributeName	required

The implementation of the `IdentifyingAttributes` element is shown in Figure 123.

```
<xs:element name="identifyingAttributes" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="attribute" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="name" type="XDM_AttributeName" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Figure 123.Implementation of the `IdentifyingAttributes` element

Element: Sequencers



The `Sequnecers` element defines the names of sequences used during import to generate values of the table key attributes. Each `Sequencer` child element defines a single mapping between a database attribute and a database sequence.

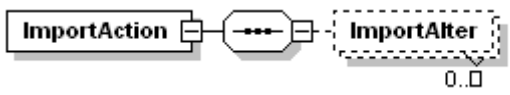
Name	Type	Use
att	XDM_AttributeName	required
seq	xs:string	required

The implementation of the `Sequencers` element is shown in Figure 124.

```
<xs:element name="Sequencers" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Sequencer" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="att" type="XDM_AttributeName" use="required"/>
          <xs:attribute name="seq" type="xs:string" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Figure 124. Implementation of the `Sequencers` element

Element: ImportAction



The `ImportAction` element defines the default action undertaken during the import process of the XDE element. `ImportAlter` child elements define which attributes have to be changed before the import action. Depending on value of the `action` attribute of the `ImportAction` element the following procedures may be performed by the importer:

- `UseOld` – the importer will try to find in the target database the row having the same attribute values as defined in the `IdentifyingAttributes` XDM element,
- `AddNew` – the importer will import data using the sequences defined in the `Sequencers` XDM element,

- `Skip` – the XDE element will be ignored in the import,
- `AskUser` – a user will be asked to decide which one of the above actions should be performed.

Attributes:

Name	Type	Use
action	XDM_ImportActions	required

Child element: `ImportAlter`

Attributes:

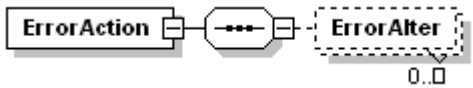
Name	Type	Use
att	XDM_AttributeName	required

The implementation of the `ImportAction` element is shown in Figure 125.

```
<xs:element name="ImportAction" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ImportAlter" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="att" type="XDM_AttributeName" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="action" type="XDM_ImportActions" use="required"/>
  </xs:complexType>
</xs:element>
```

Figure 125.Implementation of the `ImportAction` element

Element: ErrorAction



The `ErrorAction` element defines the action that should be undertaken when default import action fails. The `ErrorAlter` child elements define which attributes have to be changed before re-attempting to import the data. Depending on value of the `action` attribute of the `ErrorAction` element different procedures may be performed by the importer.

Attributes:

Name	Type	Use
action	XDM_ImportActions	required

Child element: `ErrorAlter`

Attributes:

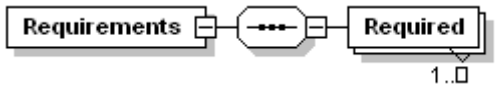
Name	Type	Use
att	XDM_AttributeName	required

The implementation of the `ErrorAction` element is shown in Figure 126.

```
<xs:element name="ErrorAction" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ErrorAlter" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="att" type="XDM_AttributeName" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="action" type=" XDM_ImportActions " use="required"/>
  </xs:complexType>
</xs:element>
```

Figure 126.Implementation of the `ErrorAction` element

Element: Requirements



The `Requirements` element is used when a key relationship in the database is modelled as a 'parent-child' hierarchy in the XDE schema. The element provides information about the foreign key (and corresponding primary key) attributes connecting the database entities corresponding to the parent and the child elements in XDE. In the `Required` child element, the foreign key database attribute (`att`) of to the child element and the corresponding primary key database attribute (`parentAtt`) of the parent element are defined.

Name	Type	Use
att	XDM_AttributeName	required
parentAtt	XDM_AttributeName	required

The implementation of the `Requirements` element is shown in Figure 127.

```
<xs:element name="Requirements" minOccurs="0">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Required" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="att" type="XDM_AttributeName" use="required"/>
          <xs:attribute name="parentAtt" type="XDM_AttributeName" use="required"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Figure 127.Implementation of the `Requirements` element

4.5.4. Example XDE Element

An example of using XDM elements in the XDE schema of A_ASSOC_PARAM_VALUE element is presented in Figure 128.

The A_ASSOC_PARAM_VALUE element corresponds to the A_ASSOC_PARAM_VALUES table in the ARCO database. The default import action is to update the existing database row. If this operation fails, the new row should be created using A_VALUES_SEQ sequencer. The MO_ID parent attribute value is used in the export and import as value of the APV_MO_ID attribute of A_ASSOC_PARAM_VALUE element.

```
<xs:element name="A_ASSOC_PARAM_VALUE" minOccurs="0" maxOccurs="unbounded">
   <xs:annotation>
      <xs:documentation>This element contains .... </xs:documentation>
      <xs:appinfo>
         <XDM>
            <Map name="A_ASSOC_PARAM_VALUES"/>
            <Type value="DatabaseRelatedElement"/>
            <ImportAction action="UseOld"/>
            <ErrorAction action="AddNew"/>
            <Requirements>
               <Required att="APV_MO_ID" parentAtt="MO_ID"/>
            </Requirements>
            <Sequencers>
               <Sequencer att="APV_ID" seq="A_VALUES_SEQ"/>
            </Sequencers>
         </XDM>
      </xs:appinfo>
   </xs:annotation>
   <xs:complexType>
      <xs:choice>
         <xs:element name="APV_VALUE_VARCHAR2" type="xde:AT_SHORT_STRING">
            <xs:annotation>
               <xs:appinfo>
                  <XDM>
                     <Type value="ElementAsAttribute"/>
                  </XDM>
               </xs:appinfo>
            </xs:annotation>
         </xs:element>

            ...

         </xs:choice>
      <xs:attribute name="APV_AP_ID" type="xde:AT_ID" use="required"/>
   </xs:complexType>
   <xs:keyref name="APV_AP_ID" refer="xde:AP_ID">
      <xs:selector xpath="."/>
      <xs:field xpath="@APV_AP_ID"/>
   </xs:keyref>
</xs:element>
```

Figure 128.Example of using XDM – A_ASSOC_PARAM_VALUES XDE element

4.6. XDE Schema Overview

The XDE schema contains a list the XDE data type definitions and a declaration of one root-element: ARCO_DATA. The XDE data type definitions and the overall structure of the ARCO_DATA element are discussed in this section. Details on implementation of the ARCO_DATA element can be found in Sections 4.7 and 4.8.

4.6.1. XDE Data Types

The following data types are defined in XDE:

- AT_ID – used for attributes corresponding to numeric primary and foreign keys in the database. The AT_ID type is a restriction of the xs:long type (where xs="http://www.w3.org/2001/XMLSchema");
- AT_CODE – used for attributes corresponding to string primary and foreign keys in the database. The AT_CODE is a restriction of the xs:string type with max length set to 15 characters;
- AT_DATE – used for attributes containing date and time values. It is a restriction of the xs:dataTime type;
- AT_SHORT_STRING – general text type; used for short text elements and attributes; defined as a restriction of xs:string type with max length set to 2000 characters;
- AT_VERSION – used for attributes containing version description. It is defined as a restriction of xde:AT_SHORT_STRING type with max length set to 200 characters;
- AT_DATA_TYPE_PATTERN – used for attributes containing data type patterns; defined as a restriction of xde:AT_SHORT_STRING type;
- AT_MT_EXTENSIONS – used for attributes containing file extensions for MIME-types; defined as a restriction of xde:AT_SHORT_STRING type;
- AT_PASSWORD – used for attributes containing user passwords. It is defined as a restriction of xs:hexBinary type. Elements of AT_PASSWORD type carry data in an encrypted format;
- AT_NAME – general text type; used for attributes containing names. It is defined as a restriction of xde:AT_SHORT_STRING type with max length set to 250 characters;
- AT_DESCRIPTION – used for attributes containing descriptions. It is defined as a restriction of xde:AT_SHORT_STRING type;
- AT_LABEL – used for attributes containing label names; defined as a restriction of xde:AT_SHORT_STRING;
- AT_FILENAME – used for attributes containing file location paths. It is defined as a restriction of xs:anyURI type with max length set to 500 characters;
- AT_CO_TYPE – used for attributes carrying Cultural Object types. It is defined as an enumeration type with two possible values:
  - ACQUIRED
  - REFINED
- AT_MO_NATURE_TYPE – used for attributes carrying information about nature (method of storage) of Media Objects. It is defined as an enumeration type with two possible values:
  - TEXT

    - BINARY

- AT_TPL_TYPE – used for attributes providing information about X-VRML template types. It is defined as an enumeration type with two possible values:
    - 3D
    - 2D
- AT_DT_VALUE_CATEGORY – used for attributes carrying value category of data type. It is defined as an enumeration type with two possible values:
    - DIRECT_VALUE
    - GENERAL_DATA
- AT_TEMPLATE_PARAMETER_VALUE_CATEGORY – used for attributes carrying category of X-VRML template parameter values. It is defined as an enumeration type with several possible values:
    - TEMPLATE_OBJECT
    - GENERAL_DATA
    - DIRECT_VALUE
    - MEDIA_OBJECT
    - TEMPLATE
    - MEDIA_OBJECT_TYPE
    - TEMPLATE_INSTANCE
    - XSL_TEMPLATE
- AT_STRING – general text type; used for text elements with unrestricted length; defined as a restriction of the xs:string type;
- AT_XML_AMS – used for elements containing metadata structured according to AMS – ARCO Metadata Schema. It is defined as a restriction of xs:any type with namespace set to "##any"; The namespace may be restricted in future prototypes.
- AT_XML_XSD – used for elements containing XML Schemas. It is defined as a restriction of xs:any type with namespace set to "##any"; The namespace may be restricted in future prototypes.
- AT_XML_X-VRML – used for elements containing X-VRML Templates. It is defined as a restriction of xs:any type with namespace set to "##any"; The namespace may be restricted in future prototypes.
- AT_XML_XSL – used for elements containing XML stylesheets. It is defined as a restriction of xs:any type with namespace set to "##any"; The namespace may be restricted in future prototypes.
- AT_BINARY – used for elements containing binary data; defined as a restriction of xs:base64Binary type;
- AT_PRIMARY_CONTENTS – used for attributes carrying description of XDE file primary contents. It is defined as an enumeration type with several possible values:
    - CULTURAL_OBJECT – XDE file contains Cultural Objects, Cultural Object Folders, Media Objects, and all related data;
    - MEDIA_OBJECT – XDE file contains a single Media Object with all related data (sub-Media Objects, metadata, etc.);

  - ARIF – XDE file contains ARIF Folders, Cultural Object Folders, Template Instances, Template Objects, and all other related data;
  - TEMPLATE – XDE file contains Templates, Template Instances, Template Object Types, Template Objects, and all other related data;
  - DEFINITIONS – XDE file contains rarely changing data such as AMS XML Schemas, Media Object Types, Data Types, and MIME-Types;
  - DATABASE – XDE file contains selection of the whole ARCO database contents.
- AT_DATA_SOURCE – used for attributes containing information about the data source of the XDE file content. It is defined as an enumeration type with several possible values:
  - OM
  - IMRR
  - DATABASE
  - OTHER
- AT_LEVEL – used for PRO_LEVEL attribute of the PRO element. It is defined as a restriction of xde:AT_SHORT_STRING;
- AT_PRO_CATEGORY – used for PRO_CATEGORY attribute of the PRO element. It is defined as an enumeration type with several possible values:
  - DIRECT_VALUE
  - GENERAL_DATA
  - TEMPLATE_OBJECTS
  - XSL_TEMPLATE
- AT_TPL_NATURE – used for TPL_NATURE attribute of the XVRML_TEMPLATE element. It is defined as an enumeration type with several possible values:
  - PROC
  - TPL
  - CLASS
  - MODULE

4.6.2. XDE Data Structure

The root element of the XDE file is ARCO_DATA. The overall structure of the ARCO_DATA element is presented in the Figure 129. It contains two child elements corresponding to the two logical parts of the XDE content:

- The DYNAMIC_DATA element – used to carry the actual data being exchanged, e.g. Cultural Objects, Media Objects, Templates, and Template Instances;
- The STATIC_DATA element – used to carry rarely changing data. Examples are: MIME-types, Media Object Types, Data Types, and System Configuration Data.

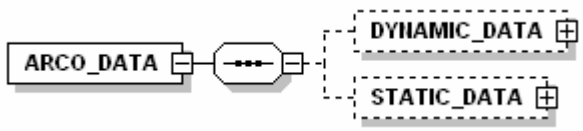


Figure 129.XDE root element - ARCO_DATA

The DYNAMIC_DATA and STATIC_DATA elements are described in Sections 4.7 and 4.8, respectively.

4.7. ARCO Dynamic Data

The structure of the DYNAMIC_DATA element is shown in the Figure 130.

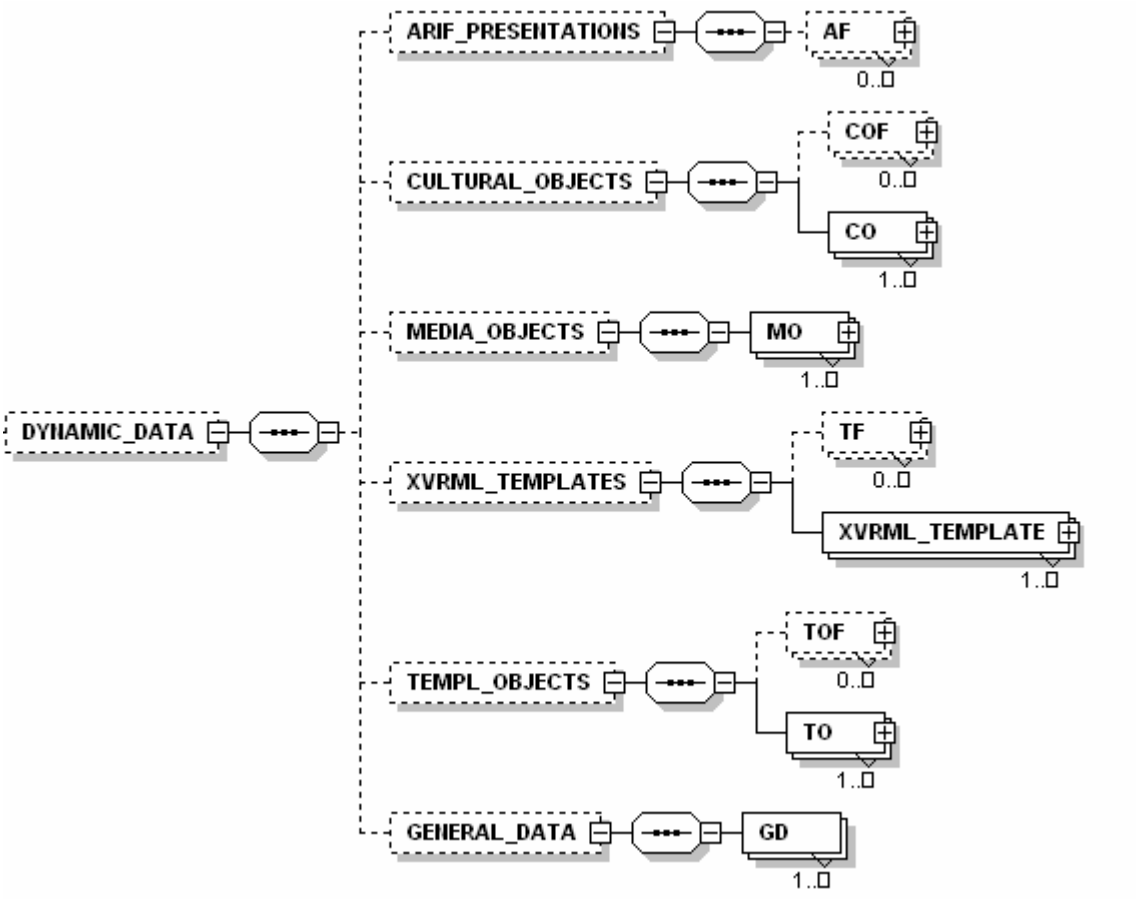


Figure 130.The structure of the DYNAMIC_DATA element

4.7.1. Element AF – ARIF Folder

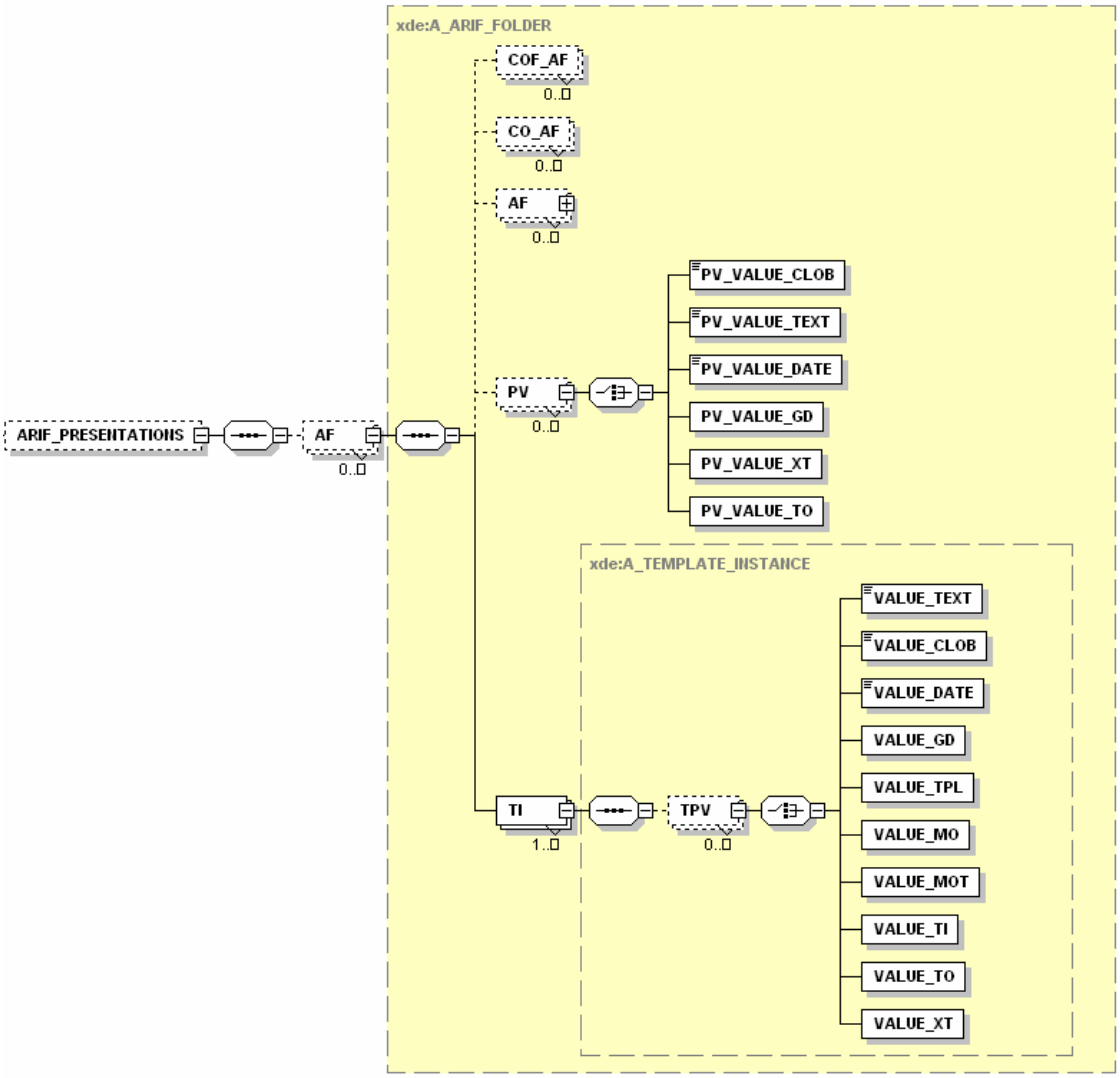


Figure 131.XML Schema of the AF element

The AF element represents the ARIF Folder (Figure 131). The COF_AF child element corresponds to the A_CO_FOLDER_IN_ARIF_FOLDERS table and has a key reference to Cultural Object Folder. The CO_AF child element corresponds to the A_CULTURAL_OBJ_IN_ARIF_FOLDER table and has key reference to the Cultural Object. The AF child element represents ARIF subfolders. The PV child element corresponds the A_PROPERTY_VALUES table. The TI child element corresponds to the A_TEMPLATE_INSTACES table.

Attributes:

Name	Type	Use
AF_ID	xde:AT_ID	required
AF_NAME	xde:AT_NAME	required
AF_DESCRIPTION	xde:AT_DESCRIPTION	optional

Constraints:

Name	Refer	Field(s)

key	AF_ID		@AF_ID

Child element: COF_AF

Attributes:

Name	Type	Use
COAF_AF_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	COAF_AF_ID	xde:COF_ID	@CFAF_AF_ID

Child element: CO_AF

Attributes:

Name	Type	Use
COAF_AF_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	COAF_AF_ID	xde:CO_ID	@CAOF_AF_ID

Child element: PV

Attributes:

Name	Type	Use
PV_ID	xde:AT_ID	required
PV_PRO_ID	xde:AT_ID	required
PV_CO_ID	xde:AT_ID	optional
PV_MO_ID	xde:AT_ID	optional

Constraints:

	Name	Refer	Field(s)
key	PV_ID		@PV_ID
keyref	PV_PRO_ID	xde:PRO_ID	@PV_PRO_ID
keyref	PV_CO_ID	xde:CO_ID	@PV_CO_ID
keyref	PV_MO_ID	xde:MO_ID	@PV_MO_ID

Child element: TI

Attributes:

Name	Type	Use
TI_ID	xde:AT_ID	required
TI_NAME	xde:AT_NAME	required
TI_DESCRIPTION	xde:AT_DESCRIPTION	optional

TI_TPL_ID	xde:AT_ID	required
TI_TD_ID	xde:AT_ID	optional

Constraints:

	Name	Refer	Field(s)
key	TI_ID		@TI_ID
keyref	TI_TPL_ID	xde:TPL_ID	@TI_TPL_ID
keyref	TI_TD_ID	xde:TD_ID	@TI_TD_ID

The implementation of the AF element is shown in Figure 132, while its type definition in Figure 133. The A_TEMPLATE_INSTANCE type definition is shown in Figure 134.

```
<xs:element name="AF" type="xde:A_ARIF_FOLDER" minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_ARIF_FOLDERS"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="AF_ID" seq="A_FOLDERS_SEQ"/>
        </Sequencers>
        <Requirements>
          <Required att="AF_PARENT_FOLD_ID" parentAtt="AF_ID"/>
        </Requirements>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:key name="AF_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@AF_ID"/>
  </xs:key>
</xs:element>
```

Figure 132.Implementation of the AF element

```
<xs:complexType name="A_ARIF_FOLDER">
  <xs:sequence>
    <xs:element name="COF_AF" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="exp"/>
            <Map name="A_CO_FOLDERS_IN_ARIF_FOLDERS"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="CFAF_AF_ID" parentAtt="AF_ID"/>
            </Requirements>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="CFAF_COF_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
      <xs:keyref name="CFAF_COF_ID" refer="xde:COF_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@CFAF_AF_ID"/>
      </xs:keyref>
    </xs:element>
    <xs:element name="CO_AF" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="exp"/>
            <Map name="A_CULTURAL_OBJ_IN_ARIF_FOLDERS"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="COAF_AF_ID" parentAtt="AF_ID"/>
            </Requirements>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="COAF_CO_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
      <xs:keyref name="COAF_CO_ID" refer="xde:CO_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@COAF_CO_ID"/>
      </xs:keyref>
    </xs:element>
    <xs:element name="AF" type="xde:A_ARIF_FOLDER" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="PV" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="exp"/>
            <Map name="A_PROPERTY_VALUES"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Sequencers>
              <Sequencer att="PV_ID" seq="A_VALUES_SEQ"/>
            </Sequencers>
            <Requirements>
              <Required att="PV_AF_ID" parentAtt="AF_ID"/>
            </Requirements>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:choice>
          <xs:element name="PV_VALUE_CLOB" type="xde:AT_STRING">
```

```
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
          </xs:element>
          <xs:element name="PV_VALUE_TEXT" type="xde:AT_SHORT_STRING">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                  <Map name="PV_VALUE_VARCHAR2"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
          </xs:element>
          <xs:element name="PV_VALUE_DATE" type="xde:AT_DATE">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
          </xs:element>
          <xs:element name="PV_VALUE_GD">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="PV_VALUE_GD_ID" type="xde:AT_ID" use="required">
                <xs:annotation>
                  <xs:appinfo>
                    <XDM>
                      <Type value="attEl"/>
                    </XDM>
                  </xs:appinfo>
                </xs:annotation>
              </xs:attribute>
            </xs:complexType>
            <xs:keyref name="PV_VALUE_GD_ID" refer="xde:GD_ID">
              <xs:selector xpath="."/>
              <xs:field xpath="@PV_VALUE_GD_ID"/>
            </xs:keyref>
          </xs:element>
          <xs:element name="PV_VALUE_XT">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="PV_VALUE_XT_ID" type="xde:AT_ID" use="required">
                <xs:annotation>
                  <xs:appinfo>
                    <XDM>
                      <Type value="attEl"/>
                    </XDM>
                  </xs:appinfo>
```

```
            </xs:annotation>
          </xs:attribute>
        </xs:complexType>
        <xs:keyref name="PV_VALUE_XT_ID" refer="xde:XT_ID">
          <xs:selector xpath="."/>
          <xs:field xpath="@PV_VALUE_XT_ID"/>
        </xs:keyref>
      </xs:element>
      <xs:element name="PV_VALUE_TO">
        <xs:annotation>
          <xs:appinfo>
            <XDM>
              <Type value="attEl"/>
            </XDM>
          </xs:appinfo>
        </xs:annotation>
        <xs:complexType>
          <xs:attribute name="PV_VALUE_TO_ID" type="xde:AT_ID" use="required">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
          </xs:attribute>
        </xs:complexType>
        <xs:keyref name="PV_VALUE_TO_ID" refer="xde:TO_ID">
          <xs:selector xpath="."/>
          <xs:field xpath="@PV_VALUE_TO_ID"/>
        </xs:keyref>
      </xs:element>
    </xs:choice>
    <xs:attribute name="PV_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="PV_PRO_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="PV_CO_ID" type="xde:AT_ID" use="optional"/>
    <xs:attribute name="PV_MO_ID" type="xde:AT_ID" use="optional"/>
  </xs:complexType>
  <xs:key name="PV_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@PV_ID"/>
  </xs:key>
  <xs:keyref name="PV_PRO_ID" refer="xde:PRO_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@PV_PRO_ID"/>
  </xs:keyref>
  <xs:keyref name="PV_CO_ID" refer="xde:CO_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@PV_CO_ID"/>
  </xs:keyref>
  <xs:keyref name="PV_MO_ID" refer="xde:MO_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@PV_MO_ID"/>
  </xs:keyref>
</xs:element>
<xs:element name="TI" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_TEMPLATE_INSTANCES"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="TI_ID" seq="A_TEMPLATE_SEQ"/>
        </Sequencers>
        <Requirements>
```

```
              <Required att="TI_AF_ID" parentAtt="AF_ID"/>
            </Requirements>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="xde:A_TEMPLATE_INSTANCE">
            <xs:attribute name="TI_TD_ID" type="xde:AT_ID" use="optional"/>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
      <xs:key name="TI_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@TI_ID"/>
      </xs:key>
      <xs:keyref name="TI_TPL_ID" refer="xde:TPL_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@TI_TPL_ID"/>
      </xs:keyref>
      <xs:keyref name="TI_TD_ID" refer="xde:TD_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@TI_TD_ID"/>
      </xs:keyref>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="AF_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="AF_NAME" type="xde:AT_NAME" use="required"/>
  <xs:attribute name="AF_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
</xs:complexType>
```

Figure 133.Implementation of the A_ARIF_FOLDER type

```
<xs:complexType name="A_TEMPLATE_INSTANCE">
  <xs:sequence>
    <xs:element name="TPV" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="exp"/>
            <Map name="A_TEMPLATE_PARAM_VALUES"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="TPV_TI_ID" parentAtt="TI_ID"/>
            </Requirements>
            <Sequencers>
              <Sequencer att="TPV_ID" seq="A_VALUES_SEQ"/>
            </Sequencers>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:choice>
          <xs:element name="VALUE_TEXT" type="xde:AT_SHORT_STRING">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                  <Map name="TPV_VALUE_VARCHAR2"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
          </xs:element>
          <xs:element name="VALUE_CLOB" type="xde:AT_STRING">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                  <Map name="TPV_VALUE_CLOB"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
          </xs:element>
          <xs:element name="VALUE_DATE" type="xde:AT_DATE">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                  <Map name="TPV_VALUE_DATE"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
          </xs:element>
          <xs:element name="VALUE_GD">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                  <Map name="TPV_VALUE_GD"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="TPV_VALUE_GD_ID" type="xde:AT_ID" use="required"/>
            </xs:complexType>
            <xs:keyref name="TPV_VALUE_GD_ID" refer="xde:GD_ID">
              <xs:selector xpath="."/>
              <xs:field xpath="@TPV_VALUE_GD_ID"/>
            </xs:keyref>
```

```
        </xs:element>
        <xs:element name="VALUE_TPL">
          <xs:annotation>
            <xs:appinfo>
              <XDM>
                <Type value="attEl"/>
              </XDM>
            </xs:appinfo>
          </xs:annotation>
          <xs:complexType>
            <xs:attribute name="TPV_VALUE_TPL_ID" type="xde:AT_ID" use="required">
              <xs:annotation>
                <xs:appinfo>
                  <XDM>
                    <Type value="attEl"/>
                  </XDM>
                </xs:appinfo>
              </xs:annotation>
            </xs:attribute>
          </xs:complexType>
          <xs:keyref name="TPV_VALUE_TPL_ID" refer="xde:TPL_ID">
            <xs:selector xpath="."/>
            <xs:field xpath="@TPV_VALUE_TPL_ID"/>
          </xs:keyref>
        </xs:element>
        <xs:element name="VALUE_MO">
          <xs:annotation>
            <xs:appinfo>
              <XDM>
                <Type value="attEl"/>
              </XDM>
            </xs:appinfo>
          </xs:annotation>
          <xs:complexType>
            <xs:attribute name="TPV_VALUE_MO_ID" type="xde:AT_ID" use="required">
              <xs:annotation>
                <xs:appinfo>
                  <XDM>
                    <Type value="attEl"/>
                  </XDM>
                </xs:appinfo>
              </xs:annotation>
            </xs:attribute>
          </xs:complexType>
          <xs:keyref name="TPV_VALUE_MO_ID" refer="xde:MO_ID">
            <xs:selector xpath="."/>
            <xs:field xpath="@TPV_VALUE_MO_ID"/>
          </xs:keyref>
        </xs:element>
        <xs:element name="VALUE_MOT">
          <xs:annotation>
            <xs:appinfo>
              <XDM>
                <Type value="attEl"/>
              </XDM>
            </xs:appinfo>
          </xs:annotation>
          <xs:complexType>
            <xs:attribute name="TPV_VALUE_MOT_ID" type="xde:AT_ID" use="required">
              <xs:annotation>
                <xs:appinfo>
                  <XDM>
                    <Type value="attEl"/>
                  </XDM>
                </xs:appinfo>
              </xs:annotation>
            </xs:attribute>
```

```
            </xs:complexType>
            <xs:keyref name="TPV_VALUE_MOT_ID" refer="xde:MOT_ID">
              <xs:selector xpath="."/>
              <xs:field xpath="@TPV_VALUE_MOT_ID"/>
            </xs:keyref>
          </xs:element>
          <xs:element name="VALUE_TI">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="TPV_VALUE_TI_ID" type="xde:AT_ID" use="required">
                <xs:annotation>
                  <xs:appinfo>
                    <XDM>
                      <Type value="attEl"/>
                    </XDM>
                  </xs:appinfo>
                </xs:annotation>
              </xs:attribute>
            </xs:complexType>
            <xs:keyref name="TPV_VALUE_TI_ID" refer="xde:TI_ID">
              <xs:selector xpath="."/>
              <xs:field xpath="@TPV_VALUE_TI_ID"/>
            </xs:keyref>
          </xs:element>
          <xs:element name="VALUE_TO">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="TPV_VALUE_TO_ID" type="xde:AT_ID" use="required">
                <xs:annotation>
                  <xs:appinfo>
                    <XDM>
                      <Type value="attEl"/>
                    </XDM>
                  </xs:appinfo>
                </xs:annotation>
              </xs:attribute>
            </xs:complexType>
            <xs:keyref name="TPV_VALUE_TO_ID" refer="xde:TO_ID">
              <xs:selector xpath="."/>
              <xs:field xpath="@TPV_VALUE_TO_ID"/>
            </xs:keyref>
          </xs:element>
          <xs:element name="VALUE_XT">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="TPV_VALUE_XT_ID" type="xde:AT_ID" use="required">
                <xs:annotation>
                  <xs:appinfo>
                    <XDM>
```

```
                        <Type value="attEl"/>
                      </XDM>
                    </xs:appinfo>
                  </xs:annotation>
                </xs:attribute>
              </xs:complexType>
              <xs:keyref name="TPV_VALUE_XT_ID" refer="xde:XT_ID">
                <xs:selector xpath="."/>
                <xs:field xpath="@TPV_VALUE_XT_ID"/>
              </xs:keyref>
            </xs:element>
          </xs:choice>
          <xs:attribute name="TPV_TP_ID" type="xde:AT_ID" use="required"/>
        </xs:complexType>
        <xs:keyref name="TPV_TP_ID" refer="xde:TP_ID">
          <xs:selector xpath="."/>
          <xs:field xpath="@TPV_TP_ID"/>
        </xs:keyref>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="TI_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="TI_NAME" type="xde:AT_NAME" use="required"/>
    <xs:attribute name="TI_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
    <xs:attribute name="TI_TPL_ID" type="xde:AT_ID" use="required"/>
</xs:complexType>
```

Figure 134.Implementation of the A_TEMPLATE_INSTANCE type

4.7.2. Element COF – Cultural Object Folder

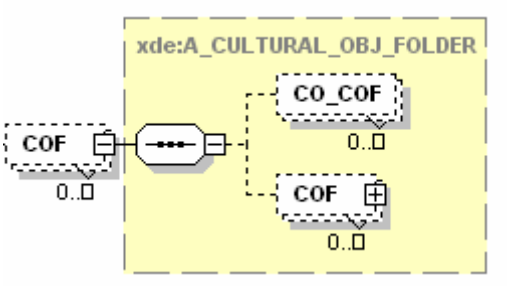


Figure 135.XML Schema of the COF element

The `COF` element corresponds to the `A_CULTURAL_OBJ_FOLDERS` table, and represents the hierarchical structure of Cultural Object Folders (Figure 135). The optional and unbounded element `CO_COF` corresponds to the `A_CULTURAL_OBJ_IN_FOLDER` table and has a key reference to the Cultural Object assigned to this folder. The optional and unbounded element `COF` represents subfolders assigned to this folder.

Attributes:

Name	Type	Use
COF_ID	xde:AT_ID	required
COF_NAME	xde:AT_NAME	required
COF_DESCRIPTION	xde:AT_DESCRIPTION	optional
COF_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
key	COF_ID		@COF_ID
keyref	COF_PARENT_FOLDER_ID	xde:COF_ID	@COF_PARENT_FOLDER_ID

Child element: `CO_COF`

Attributes:

Name	Type	Use
COIF_CO_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	COIF_CO_ID	xde:CO_ID	@COIF_CO_ID

The implementation of the `COF` element is shown in Figure 136, while its type definition in Figure 137.

```
<xs:element name="COF" minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_CULTURAL_OBJ_FOLDERS"/>
        <ImportAction action="addNew">
          <ImportAlter att="COF_NAME"/>
        </ImportAction>
        <ErrorAction action="addNew">
          <ErrorAlter att="COF_NAME"/>
        </ErrorAction>
        <IdentifyingAttributes>
          <Attribute name="COF_NAME"/>
        </IdentifyingAttributes>
        <Sequencers>
          <Sequencer att="COF_ID" seq="A_FOLDERS_SEQ"/>
        </Sequencers>
        <Requirements>
          <Required att="COF_PARENT_FOLDER_ID" parentAtt="COF_ID"/>
        </Requirements>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="xde:A_CULTURAL_OBJ_FOLDER"/>
    </xs:complexContent>
  </xs:complexType>
  <xs:key name="COF_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@COF_ID"/>
  </xs:key>
  <xs:keyref name="COF_PARENT_FOLDER_ID" refer="xde:COF_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@COF_PARENT_FOLDER_ID"/>
  </xs:keyref>
</xs:element>
```

Figure 136.Implementation of the COF element

```
<xs:complexType name="A_CULTURAL_OBJ_FOLDER">
  <xs:annotation>
    <xs:appinfo/>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="CO_COF" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Map name="A_CULTURAL_OBJ_IN_FOLDERS"/>
            <Type value="exp"/>
            <ImportAction action="addNew"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="COIF_COF_ID" parentAtt="COF_ID"/>
            </Requirements>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="COIF_CO_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
      <xs:keyref name="COIF_CO_ID" refer="xde:CO_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@COIF_CO_ID"/>
      </xs:keyref>
    </xs:element>
    <xs:element name="COF" type="xde:A_CULTURAL_OBJ_FOLDER" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo/>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="COF_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="COF_NAME" type="xde:AT_NAME" use="required"/>
  <xs:attribute name="COF_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
</xs:complexType>
```

Figure 137.Implementation of the A_CULTURAL_OBJ_FOLDER type

4.7.3. Element CO – Cultural Object

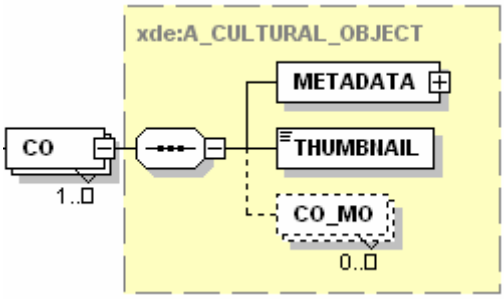


Figure 138.XML Schema of the A_CULTURAL_OBJECT element

The `CO` element corresponds to the `A_CULTURAL_OBJECTS` table and represents Cultural Objects from the ARCO database (Figure 138). The optional and unbounded child element `CO_MO` corresponds to the `A_CULTURAL_OBJ_TO_MEDIA_OBJ` table and has a key reference to the Media Object. This represents a list of all Media Object associated with this Cultural Object. The `METADATA` element is an XML document containing Cultural Object AMS metadata together with either Acquired Object or Refined Object AMS metadata depending on the `AT_CO_TYPE` attribute value. The child element `THUMBNAIL` contains the Cultural Object thumbnail picture.

Attributes:

Name	Type	Use
CO_ID	xde:AT_ID	required
CO_NAME	xde:AT_NAME	required
CO_CREATED_ON	xde:AT_DATE	required
CO_CREATED_FROM_CO_ID	xde:AT_ID	required
CO_CMSV_ID	xde:AT_ID	required
AT_CO_TYPE	xde:AT_CO_TYPE	required

Constraints:

	Name	Refer	Field(s)
key	CO_ID		@CO_ID
keyref	CO_CMSV_ID	xde:CMSV_ID	@CO_CMSV_ID
keyref	CO_CREATED_FROM_CO_ID	xde:CO_ID	@CO_CREATED_FROM_CO_ID

Child element: `CO_MO`

Attributes:

Name	Type	Use
COMO_MO_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	COMO_MO_ID	xde:MO_ID	@COMO_MO_ID

The implementation of the CO element is shown in Figure 139, while its type definition in Figure 140.

```
<xs:element name="CO" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_CULTURAL_OBJECTS"/>
        <ImportAction action="addNew"/>
        <ErrorAction action="addNew">
          <ErrorAlter att="CO_NAME"/>
        </ErrorAction>
        <IdentifyingAttributes>
          <Attribute name="CO_NAME"/>
        </IdentifyingAttributes>
        <Sequencers>
          <Sequencer att="CO_ID" seq="A_OBJECTS_SEQ"/>
        </Sequencers>
        <Requirements>
          <Required att="CO_CREATED_FROM_CO_ID" parentAtt="CO_ID"/>
        </Requirements>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="xde:A_CULTURAL_OBJECT"/>
    </xs:complexContent>
  </xs:complexType>
  <xs:key name="CO_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@CO_ID"/>
  </xs:key>
  <xs:keyref name="CO_CMSV_ID" refer="xde:CMSV_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@CO_CMSV_ID"/>
  </xs:keyref>
  <xs:keyref name="CO_CREATED_FROM_CO_ID" refer="xde:CO_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@CO_CREATED_FROM_CO_ID"/>
  </xs:keyref>
</xs:element>
```

Figure 139.Implementation of the CO element

```
<xs:complexType name="A_CULTURAL_OBJECT">
  <xs:annotation>
    <xs:appinfo>map:A_CULTURAL_OBJECT=A_CULTURAL_OBJECTS</xs:appinfo>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="METADATA" type="xde:AT_XML_AMS">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="attEl"/>
            <Map name="CO_METADATA"/>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
    <xs:element name="THUMBNAIL" type="xde:AT_BINARY">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="attEl"/>
            <Map name="CO_THUMBNAIL"/>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
    <xs:element name="CO_MO" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="exp"/>
            <Map name="A_CULTURAL_OBJ_TO_MEDIA_OBJ"/>
            <ImportAction action="addNew"/>
            <ErrorAction action="skip"/>
            <Requirements>
              <Required att="COMO_CO_ID" parentAtt="CO_ID"/>
            </Requirements>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="COMO_MO_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
      <xs:keyref name="COMO_MO_ID" refer="xde:MO_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@COMO_MO_ID"/>
      </xs:keyref>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="CO_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="CO_NAME" type="xde:AT_NAME" use="required"/>
  <xs:attribute name="CO_CREATED_ON" type="xde:AT_DATE" use="required"/>
  <xs:attribute name="CO_CREATED_FROM_CO_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="CO_CMSV_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="CO_TYPE" type="xde:AT_CO_TYPE" use="required"/>
</xs:complexType>
```

Figure 140.Implementation of the A_CULTURAL_OBJECT type

4.7.4. Element MO – Media Object

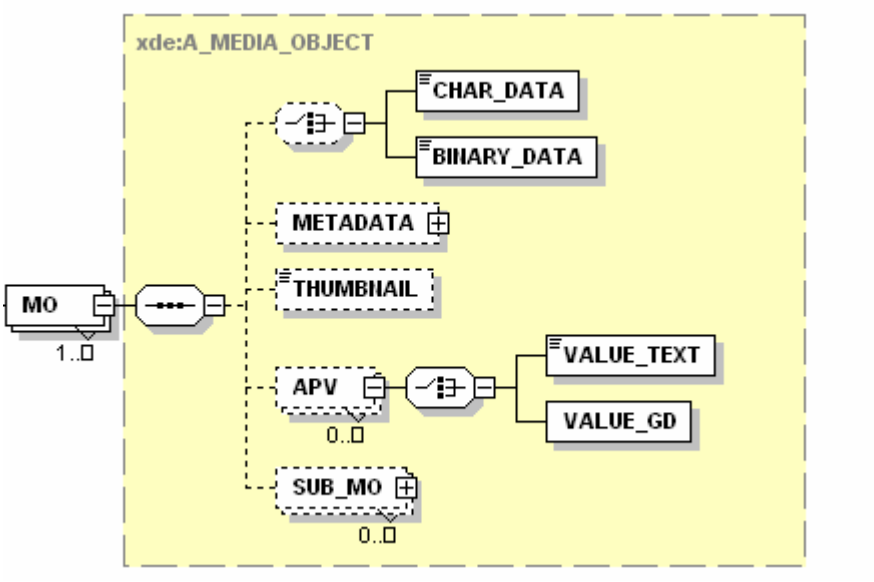


Figure 141. XML Schema of the MO element

The `MO` element corresponds to the `A_MEDIA_OBJECTS` table and represents the Media Object from the ARCO database (Figure 141). Its structure reflects the structure of composite Media Objects by joining Media Objects into a hierarchy with associations. The THUMBNAIL element contains Media Object thumbnail image. The `METADATA` element is the XML document containing Media Object AMS metadata. The actual contents of the Media Object is kept either in a textual form in the `CHAR_DATA` element, or in a binary form in the `BINARY_DATA` element, depending on the media type. The list of optional and unbounded `SUB_MO` elements joins other Media Objects to this Media Object as its sub-objects. An `MO` element may contain an unbounded list of `APV` elements, which provide values of parameters of the association between the parent Media Object and the sub-Media Objects. Each `APV` has a key reference to the appropriate `ASSOCP` definition, and a value. The `VALUE_TEXT` contains direct textual value, while the `VALUE_GD` has key reference to `GD` element.

Attributes:

Name	Type	Use
MO_ID	xde:AT_ID	required
MO_NAME	xde:AT_NAME	required
MO_CREATED_ON	xde:AT_DATE	required
MO_FILENAME	xde:AT_FILENAME	optional
MO_MOT_ID	xde:AT_ID	required
MO_MT_ID	xde:AT_ID	required
MO_MMSV_ID	xde:AT_ID	required
MO_ASS_ID	xde:AT_ID	optional
MO_MGSV_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
key	MO_ID		@MO_ID
keyref	MO_MOT_ID	xde:MOT_ID	@MO_MOT_ID
keyref	MO_MT_ID	xde:MT_ID	@MO_MT_ID
keyref	MO_MMSV_ID	xde:MMSV_ID	@MO_MMSV_ID
keyref	MO_ASS_ID	xde:ASS_ID	@MO_ASS_ID
keyref	MO_MGSV_ID	xde:MGSV_ID	@MO_MGSV_ID

Child element: APV

Attributes:

Name	Type	Use
APV_AP_ID	xde:AT_ID	required
APV_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	APV_AP_ID	xde:AP_ID	@APV_AP_ID
key	APV_ID		@APV_ID

Child element: APV/VALUE_GD

Attributes:

Name	Type	Use
APV_VALUE_GD_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	APV_VALUE_GD_ID	xde:GD_ID	@APV_VALUE_GD_ID

The implementation of the MO element is shown in Figure 142, while its type definition in Figure 143.

```
<xs:element name="MO" type="xde:A_MEDIA_OBJECT" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Map name="A_MEDIA_OBJECTS"/>
        <Type value="exp"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Requirements>
          <Required att="MO_PARENT_MO_ID" parentAtt="MO_ID"/>
        </Requirements>
        <Sequencers>
          <Sequencer att="MO_ID" seq="A_OBJECTS_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:key name="MO_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@MO_ID"/>
  </xs:key>
  <xs:keyref name="MO_MOT_ID" refer="xde:MOT_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@MO_MOT_ID"/>
  </xs:keyref>
  <xs:keyref name="MO_MT_ID" refer="xde:MT_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@MO_MT_ID"/>
  </xs:keyref>
  <xs:keyref name="MO_MMSV_ID" refer="xde:MMSV_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@MO_MMSV_ID"/>
  </xs:keyref>
  <xs:keyref name="MO_ASS_ID" refer="xde:ASS_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@MO_ASS_ID"/>
  </xs:keyref>
  <xs:keyref name="MO_MGSV_ID" refer="xde:MGSV_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@MO_MGSV_ID"/>
  </xs:keyref>
</xs:element>
```

Figure 142. Implementation of the MO element

```
<xs:complexType name="A_MEDIA_OBJECT">
  <xs:annotation>
    <xs:appinfo>map:A_MEDIA_OBJECT=A_MEDIA_OBJECTS</xs:appinfo>
  </xs:annotation>
  <xs:sequence>
    <xs:choice minOccurs="0">
      <xs:element name="CHAR_DATA" type="xde:AT_SHORT_STRING">
        <xs:annotation>
          <xs:appinfo>
            <XDM>
              <Type value="attEl"/>
              <Map name="MO_CHAR_DATA"/>
            </XDM>
          </xs:appinfo>
        </xs:annotation>
      </xs:element>
      <xs:element name="BINARY_DATA" type="xde:AT_BINARY">
        <xs:annotation>
          <xs:appinfo>
            <XDM>
              <Type value="attEl"/>
              <Map name="MO_BINARY_DATA"/>
            </XDM>
          </xs:appinfo>
        </xs:annotation>
      </xs:element>
    </xs:choice>
    <xs:element name="METADATA" type="xde:AT_XML_AMS" minOccurs="0">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="attEl"/>
            <Map name="MO_METADATA"/>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
    <xs:element name="THUMBNAIL" type="xde:AT_BINARY" minOccurs="0">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="attEl"/>
            <Map name="MO_THUMBNAIL"/>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
    <xs:element name="APV" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Map name="A_ASSOC_PARAM_VALUES"/>
            <Type value="exp"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="APV_MO_ID" parentAtt="MO_ID"/>
            </Requirements>
            <Sequencers>
              <Sequencer att="APV_ID" seq="A_VALUES_SEQ"/>
            </Sequencers>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:choice>
          <xs:element name="VALUE_TEXT" type="xde:AT_SHORT_STRING">
```

```
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                  <Map name="APV_VALUE_VARCHAR2"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
          </xs:element>
          <xs:element name="VALUE_GD">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="APV_VALUE_GD_ID" type="xde:AT_ID" use="required">
                <xs:annotation>
                  <xs:appinfo>
                    <XDM>
                      <Type value="attEl"/>
                    </XDM>
                  </xs:appinfo>
                </xs:annotation>
              </xs:attribute>
            </xs:complexType>
            <xs:keyref name="APV_VALUE_GD_ID" refer="xde:GD_ID">
              <xs:selector xpath="."/>
              <xs:field xpath="@APV_VALUE_GD_ID"/>
            </xs:keyref>
          </xs:element>
        </xs:choice>
        <xs:attribute name="APV_AP_ID" type="xde:AT_ID" use="required"/>
        <xs:attribute name="APV_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
      <xs:keyref name="APV_AP_ID" refer="xde:AP_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@APV_AP_ID"/>
      </xs:keyref>
      <xs:key name="APV_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@APV_ID"/>
      </xs:key>
    </xs:element>
    <xs:element name="SUB_MO" type="xde:A_MEDIA_OBJECT" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Map name="A_MEDIA_OBJECTS"/>
            <Type value="exp"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="MO_PARENT_MO_ID" parentAtt="MO_ID"/>
            </Requirements>
            <Sequencers>
              <Sequencer att="MO_ID" seq="A_OBJECTS_SEQ"/>
            </Sequencers>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="MO_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="MO_NAME" type="xde:AT_NAME" use="required"/>
```

```
   <xs:attribute name="MO_CREATED_ON" type="xde:AT_DATE" use="required"/>
   <xs:attribute name="MO_FILENAME" type="xde:AT_FILENAME" use="optional"/>
   <xs:attribute name="MO_MOT_ID" type="xde:AT_ID" use="required"/>
   <xs:attribute name="MO_MT_ID" type="xde:AT_ID" use="required"/>
   <xs:attribute name="MO_MMSV_ID" type="xde:AT_ID" use="required"/>
   <xs:attribute name="MO_ASS_ID" type="xde:AT_ID" use="optional"/>
   <xs:attribute name="MO_MGSV_ID" type="xde:AT_ID" use="required"/>
</xs:complexType>
```

Figure 143. Implementation of the A_MEDIA_OBJECT type

4.7.5. Element TF – Template Folder

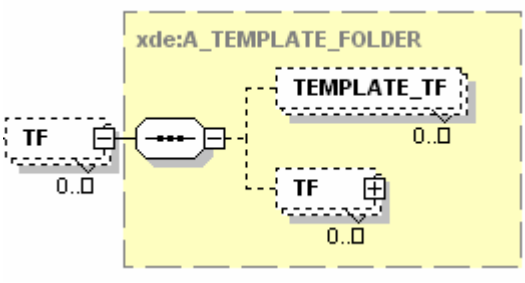


Figure 144.XML Schema of the TF element

The TF element corresponds to the A_TEMPLATE_FOLDERS table and represents template folders (Figure 144). The unbounded and optional child element TEMPLATES_TF has key reference TINF_TPL_ID pointing to the X-VRML Template. The unbounded and optional child element TF represents the list of child template folders joined to this template folder.

Attributes:

Name	Type	Use
TF_ID	xde:AT_ID	required
TF_NAME	xde:AT_NAME	required
TF_DESCRIPTION	xde:AT_DESCRIPTION	optional

Constraints:

	Name	Refer	Field(s)
key	TF_ID		@TF_ID

Child element: TEMPLATE_TF

Attributes:

Name	Type	Use
TINF_TPL_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	TINF_TPL_ID	xde:TPL_ID	@TINF_TPL_ID

The implementation of the TF element is shown in Figure 145, while its type definition in Figure 146.

```
<xs:element name="TF" minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_TEMPLATE_FOLDERS"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Requirements>
          <Required att="TF_PARENT_FOLD_ID" parentAtt="TF_ID"/>
        </Requirements>
        <Sequencers>
          <Sequencer att="TF_ID" seq="A_FOLDERS_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:complexContent>
      <xs:extension base="xde:A_TEMPLATE_FOLDER"/>
    </xs:complexContent>
  </xs:complexType>
  <xs:key name="TF_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TF_ID"/>
  </xs:key>
</xs:element>
```

Figure 145.Implementation of the TF element

```
<xs:complexType name="A_TEMPLATE_FOLDER">
  <xs:annotation>
    <xs:documentation source="required:TF_PARENT_FOLD_ID=TF_ID"/>
    <xs:documentation source="map:A_"/>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="TEMPLATE_TF" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="exp"/>
            <Map name="A_TEMPLATES_IN_FOLDERS"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="TINF_TF_ID" parentAtt="TF_ID"/>
            </Requirements>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="TINF_TPL_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
      <xs:keyref name="TINF_TPL_ID" refer="xde:TPL_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@TINF_TPL_ID"/>
      </xs:keyref>
    </xs:element>
    <xs:element name="TF" type="xde:A_TEMPLATE_FOLDER" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="TF_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="TF_NAME" type="xde:AT_NAME" use="required"/>
  <xs:attribute name="TF_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
</xs:complexType>
```

Figure 146.Implementation of the A_TEMPLATE_FOLDER type

4.7.6. Element XVRML_TEMPLATE

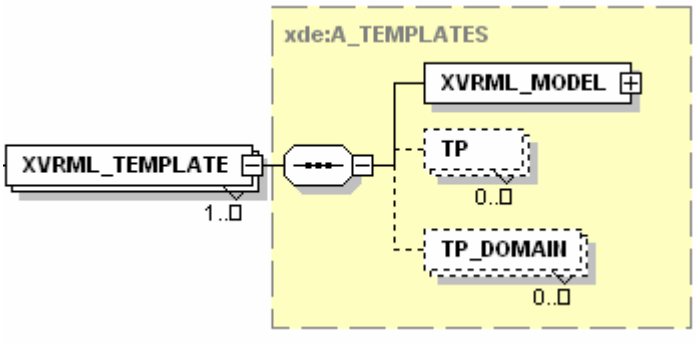


Figure 147.XML Schema of the A_TEMPLATE element

The XVRML_TEMPLATE element corresponds to the A_TEMPLATES table and represents the X-VRML Template (Figure 147). It has the child element XVRML_MODEL containing the actual XML code of the X-VRML Template definition. The optional and unbounded child element TP corresponds to the A_TEMPLATE_PARAMS table and is a list of Template parameters. The optional and unbounded child element TO_DOMAIN has reference to the template domains.

Attributes:

Name	Type	Use
TPL_ID	xde:AT_ID	required
TPL_TD_ID	xde:AT_ID	required
TPL_NAME	xde:AT_NAME	required
TPL_TYPE	xde:AT_TPL_TYPE	required
TPL_DESCRIPTION	xde:AT_DESCRIPTION	optional
TPL_SUPERIOR_TEMPL_ID	xde:AT_ID	optional
TPL_NATURE	xde:AT_TPL_NATURE	required

Constraints:

	Name	Refer	Field(s)
key	TPL_ID		@TPL_ID
keyref	TPL_SUPERIOR_TEMPL_ID	xde:TPL_ID	@TPL_SUPERIOR_TEMPL_ID

Child element: TP

Attributes:

Name	Type	Use
TP_NAME	xde:AT_NAME	required
TP_LABEL	xde:AT_LABEL	optional
TP_CATEGORY	xde:AT_TEMPLATE_PARAMETER_VALUE_CATEGORY	required
TP_DESCRIPTION	xde:AT_DESCRIPTION	optional
TP_DT_CODE	xde:AT_CODE	optional
TP_ID	xde:AT_ID	required

Constraints:

Name	Refer	Field(s)

keyref	TP_DT_CODE	xde:DT_CODE	@TP_DT_CODE
key	TP_ID		@TP_ID

Child element: TP_DOMAIN

Attributes:

Name	Type	Use
TID_TD_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	TID_TD_ID	xde:TD_ID	@TID_TD_ID

The implementation of the XVRML_TEMPLATE element is shown in Figure 148, while its type definition in Figure 149.

```
<xs:element name="XVRML_TEMPLATE" type="xde:A_TEMPLATES" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_TEMPLATES"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="TPL_ID" seq="A_TEMPLATES_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:key name="TPL_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TPL_ID"/>
  </xs:key>
  <xs:keyref name="TPL_SUPERIOR_TEMPL_ID" refer="xde:TPL_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TPL_SUPERIOR_TEMPL_ID"/>
  </xs:keyref>
</xs:element>
```

Figure 148.Implementation of the XVRML_TEMPLATE element

```
<xs:complexType name="A_TEMPLATES">
  <xs:sequence>
    <xs:element name="XVRML_MODEL" type="xde:AT_XML_X-VRML">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="attEl"/>
            <Map name="TPL_XVRML_MODEL"/>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
    <xs:element name="TP" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="exp"/>
            <Map name="A_TEMPLATE_PARAMS"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="TP_TPL_ID" parentAtt="TPL_ID"/>
            </Requirements>
            <Sequencers>
              <Sequencer att="TP_ID" seq="A_PARAMS_SEQ"/>
            </Sequencers>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="TP_NAME" type="xde:AT_NAME" use="required"/>
        <xs:attribute name="TP_LABEL" type="xde:AT_LABEL" use="optional"/>
        <xs:attribute name="TP_CATEGORY" type="xde:AT_TEMPLATE_PARAMETER_VALUE_CATEGORY"
use="required"/>
        <xs:attribute name="TP_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
        <xs:attribute name="TP_DT_CODE" type="xde:AT_CODE" use="optional"/>
        <xs:attribute name="TP_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
      <xs:keyref name="TP_DT_CODE" refer="xde:DT_CODE">
        <xs:selector xpath="."/>
        <xs:field xpath="@TP_DT_CODE"/>
      </xs:keyref>
      <xs:key name="TP_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@TP_ID"/>
      </xs:key>
    </xs:element>
    <xs:element name="TP_DOMAIN" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="exp"/>
            <Map name="A_TEMPLATES_IN_DOMAINS"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="TID_TPL_ID" parentAtt="TPL_ID"/>
            </Requirements>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="TID_TD_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
      <xs:keyref name="TID_TD_ID" refer="xde:TD_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@TID_TD_ID"/>
```

```
      </xs:keyref>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="TPL_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="TPL_NAME" type="xde:AT_NAME" use="required"/>
  <xs:attribute name="TPL_TYPE" type="xde:AT_TPL_TYPE" use="required"/>
  <xs:attribute name="TPL_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
  <xs:attribute name="TPL_SUPERIOR_TEMPL_ID" type="xde:AT_ID" use="optional"/>
  <xs:attribute name="TPL_NATURE" type="xde:AT_TPL_NATURE" use="required"/>
</xs:complexType>
```

Figure 149.Implementation of the A_TEMPLATES type

4.7.7. Element TOF – Template Object Folder

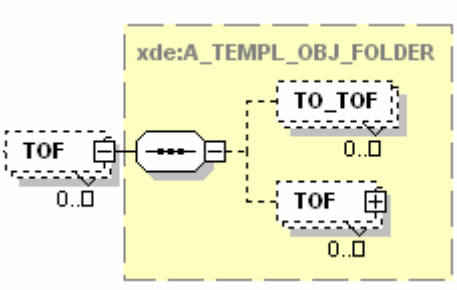


Figure 150.XML Schema of the A_TEMPL_OBJ_FOLDER element

The `TOF` element corresponds to the `A_TEMPL_OBJ_FOLDERS` table and represents Template Object Folders (Figure 150). The unbounded and optional child element `TO_TOF` has key reference `TOF_TO_ID` pointing to the Template Object.

Attributes:

Name	Type	Use
TOF_ID	xde:AT_ID	required
TOF_NAME	xde:AT_NAME	required
TOF_DESCRIPTION	xde:AT_DESCRIPTION	optional
TOF_TOF_ID	xde:AT_ID	optional

Constraints:

	Name	Refer	Field(s)
key	TOF_ID		@TOF_ID
keyref	TOF_TOF_ID	xde:TOF_ID	@TOF_TOF_ID

Child element: `TO_TOF`

Attributes:

Name	Type	Use
TOF_TO_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	TOF_TO_ID	xde:TO_ID	@TOF_TO_ID

The implementation of the `A_TEMPL_OBJ_FOLDER` element is shown in Figure 151, while its type definition in Figure 152.

```
<xs:element name="TOF" type="xde:A_TEMPL_OBJ_FOLDER" minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_TEMPL_OBJ_FOLDERS"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="TOF_ID" seq="A_FOLDERS_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:key name="TOF_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TOF_ID"/>
  </xs:key>
  <xs:keyref name="TOF_TOF_ID" refer="xde:TOF_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TOF_TOF_ID"/>
  </xs:keyref>
</xs:element>
```

Figure 151.Implementation of the TOF element

```
<xs:complexType name="A_TEMPL_OBJ_FOLDER">
  <xs:sequence>
    <xs:element name="TO_TOF" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="exp"/>
            <Map name="A_TEMPL_OBJ_IN_FOLDERS"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="TOIF_TOF_ID" parentAtt="TOF_ID"/>
            </Requirements>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="TOIF_TO_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
      <xs:keyref name="TOIF_TO_ID" refer="xde:TO_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@TOIF_TO_ID"/>
      </xs:keyref>
    </xs:element>
    <xs:element name="TOF" type="xde:A_TEMPL_OBJ_FOLDER" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="TOF_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="TOF_NAME" type="xde:AT_NAME" use="required"/>
  <xs:attribute name="TOF_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
</xs:complexType>
```

Figure 152.Implementation of the A_TEMPL_OBJ_FOLDER type

4.7.8. Element TO – Template Object

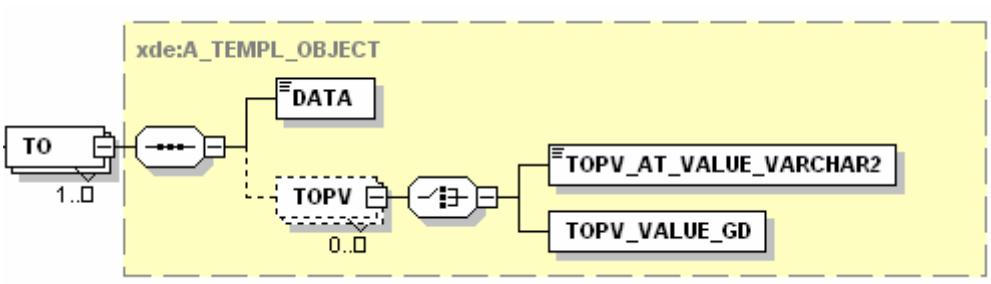


Figure 153.XML Schema of the TO element

The `TO` element corresponds to the `A_TEMPL_OBJECTS` table and represents Template Object (Figure 153). The child element `DATA` represents binary value of the Template Object, e.g. an image or a movie. The optional and unbounded element `TOPV` makes a list of template object parameters values. The `TOPV` has a key reference to the template object parameter type.

Attributes:

Name	Type	Use
TO_ID	xde:AT_ID	required
TO_NAME	xde:AT_NAME	required
TO_DESCRIPTION	xde:AT_DESCRIPTION	optional
TO_TOT_ID	xde:AT_ID	required
TO_MT_ID	xde:AT_ID	required
TO_FILENAME	xde:AT_FILENAME	optional

Constraints:

	Name	Refer	Field(s)
key	TO_ID		@TO_ID
unique	TO_NAME		@TO_NAME
keyref	TO_TOT_ID	xde:TOT_ID	@TO_TOT_ID
keyref	TO_MT_ID	xde:TD_ID	@TO_MT_ID

Child element: `TOPV`

Attributes:

Name	Type	Use
TOPV_TOTP_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	TOPV_TOTP_ID	xde:TOTP_ID	@TOPV_TOTP_ID

Child element: `TOPV_VALUE_GD`

Attributes:

Name	Type	Use
ref	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	TOPV_VALUE_GD_ID	xde:GD_ID	@ref

The implementation of the `TO` element is shown in Figure 154, while its type definition in Figure 155.

```
<xs:element name="TO" type="xde:A_TEMPL_OBJECT" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_TEMPL_OBJECTS"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="TO_ID" seq="A_OBJECTS_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:key name="TO_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TO_ID"/>
  </xs:key>
  <xs:unique name="TO_NAME">
    <xs:selector xpath="."/>
    <xs:field xpath="@TO_NAME"/>
  </xs:unique>
  <xs:keyref name="TO_TOT_ID" refer="xde:TOT_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TO_TOT_ID"/>
  </xs:keyref>
  <xs:keyref name="TO_MT_ID" refer="xde:TD_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TO_MT_ID"/>
  </xs:keyref>
</xs:element>
```

Figure 154.Implementation of the TO element

```
<xs:complexType name="A_TEMPL_OBJECT">
  <xs:sequence>
    <xs:element name="DATA" type="xde:AT_BINARY">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="attEl"/>
            <Map name="TO_DATA"/>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
    <xs:element name="TOPV" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="exp"/>
            <Map name="A_TEMPL_OBJ_PARAM_VALUES"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="TOPV_TO_ID" parentAtt="TO_ID"/>
            </Requirements>
            <Sequencers>
              <Sequencer att="TOPV_ID" seq="A_VALUES_SEQ"/>
            </Sequencers>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:choice>
          <xs:element name="TOPV_AT_VALUE_VARCHAR2" type="xde:AT_SHORT_STRING">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
          </xs:element>
          <xs:element name="TOPV_VALUE_GD">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="attEl"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="TOPV_VALUE_GD_ID" type="xde:AT_ID" use="required">
                <xs:annotation>
                  <xs:appinfo>
                    <XDM>
                      <Type value="attEl"/>
                    </XDM>
                  </xs:appinfo>
                </xs:annotation>
              </xs:attribute>
            </xs:complexType>
            <xs:keyref name="TOPV_VALUE_GD_ID" refer="xde:GD_ID">
              <xs:selector xpath="."/>
              <xs:field xpath="@TOPV_VALUE_GD_ID"/>
            </xs:keyref>
          </xs:element>
        </xs:choice>
        <xs:attribute name="TOPV_TOTP_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
```

```
      <xs:keyref name="TOPV_TOTP_ID" refer="xde:TOTP_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@TOPV_TOTP_ID"/>
      </xs:keyref>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="TO_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="TO_NAME" type="xde:AT_NAME" use="required"/>
  <xs:attribute name="TO_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
  <xs:attribute name="TO_TOT_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="TO_MT_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="TO_FILENAME" type="xde:AT_FILENAME" use="optional"/>
</xs:complexType>
```

Figure 155. Implementation of the A_TEMPL_OBJECT type

4.7.9. Element GD – General Data

Figure 156. XML Schema of the GD element

The GD element corresponds to the GENERAL_DATA_ELEMENT table and contains General Data dictionary values (Figure 156). Each GENERAL_DATA_ELEMENT element has its General Data Type indicated by a key reference GD_DT_CODE to the data type, and a value.

Attributes:

Name	Type	Use
GD_ID	xde:AT_ID	required
GD_DT_CODE	xde:AT_CODE	required
GD_VALUE	xde:AT_SHORT_STRING	optional

Constraints:

	Name	Refer	Field(s)
key	GD_ID		@GD_ID
keyref	GD_DT_CODE	xde:DT_CODE	@GD_DT_CODE

The implementation of the GENERAL_DATA_ELEMENT element is shown in Figure 157.

```
<xs:element name="GD" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Map name="A_GENERAL_DATA"/>
        <Type value="exp"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="GD_ID" seq="A_DATA_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType name="A_GENERAL_DATA">
    <xs:attribute name="GD_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="GD_DT_CODE" type="xde:AT_CODE" use="required"/>
    <xs:attribute name="GD_VALUE" type="xde:AT_SHORT_STRING" use="optional"/>
  </xs:complexType>
  <xs:key name="GD_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@GD_ID"/>
  </xs:key>
  <xs:keyref name="GD_DT_CODE" refer="xde:DT_CODE">
    <xs:selector xpath="."/>
    <xs:field xpath="@GD_DT_CODE"/>
  </xs:keyref>
</xs:element>
```

Figure 157.Implementation of the GD element

4.8. ARCO Static Data

The structure of the `STATIC_DATA` element is shown in the Figure 158.

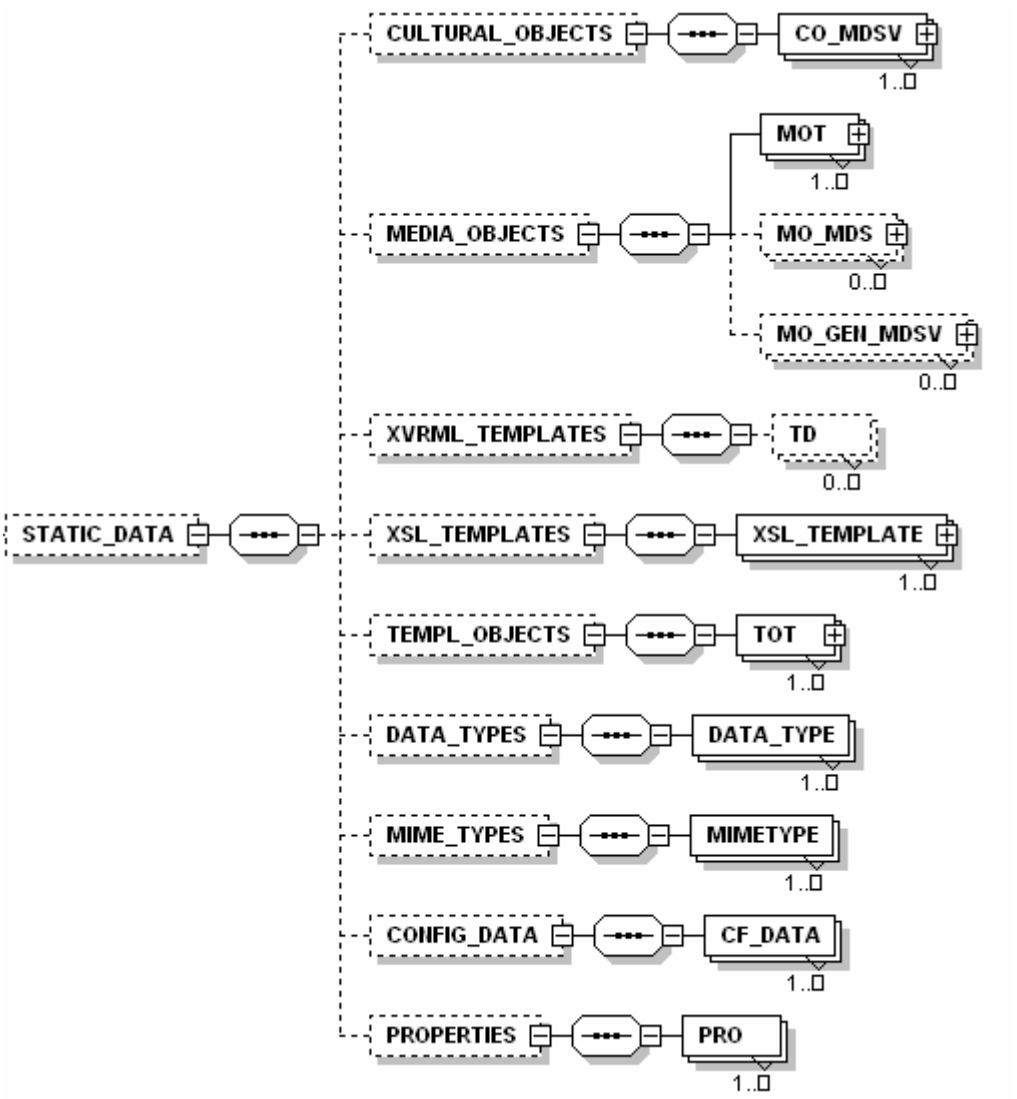


Figure 158.The structure of the STATIC_DATA element

4.8.1. Element CO_MDSV – CO Metadata Schema Version

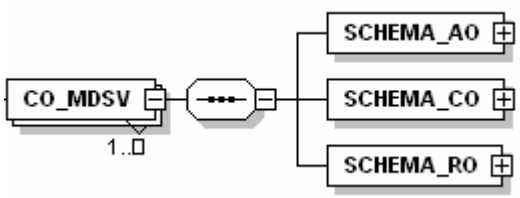


Figure 159.XML Schema of the CO_MDSV element

The CO_MDSV element corresponds to the A_CO_METDATA_SCHEMA_VERSIONS table and represents a version of Cultural Object AMS metadata schema (Figure 159). The SCHEMA_AO child element carries the Acquired Object AMS specification. The CHEMA_CO child element carries the general Cultural Object AMS specification. The CHEMA_RO child element carries the Refined Object AMS specification.

Attributes:

Name	Type	Use
CMSV_ID	xde:AT_ID	required
CMSV_ISSUED_ON	xde:AT_DATE	required
CMSV_VERSION	xde:AT_VERSION	optional

Constraints:

	Name	Refer	Field(s)
key	CMSV_ID		@CMSV_ID

The implementation of the CO_MDSV element is shown in Figure 160.

```
<xs:element name="CO_MDSV" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_CO_METADATA_SCHEMA_VERSIONS"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="CMSV_ID" seq="A_SCHEMAS_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType name="A_CO_METADATA_SCHEMA_VERSION">
    <xs:sequence>
      <xs:element name="SCHEMA_AO" type="xde:AT_XML_XSD">
        <xs:annotation>
          <xs:appinfo>
            <XDM>
              <Type value="attEl"/>
              <Map name="CMSV_SCHEMA_AO"/>
            </XDM>
          </xs:appinfo>
        </xs:annotation>
      </xs:element>
      <xs:element name="SCHEMA_CO" type="xde:AT_XML_XSD">
        <xs:annotation>
          <xs:appinfo>
            <XDM>
              <Type value="attEl"/>
              <Map name="CMSV_SCHEMA_CO"/>
            </XDM>
          </xs:appinfo>
        </xs:annotation>
      </xs:element>
      <xs:element name="SCHEMA_RO" type="xde:AT_XML_XSD">
        <xs:annotation>
          <xs:appinfo>
            <XDM>
              <Type value="attEl"/>
              <Map name="CMSV_SCHEMA_RO"/>
            </XDM>
          </xs:appinfo>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="CMSV_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="CMSV_ISSUED_ON" type="xde:AT_DATE" use="required"/>
    <xs:attribute name="CMSV_VERSION" type="xde:AT_VERSION" use="optional"/>
  </xs:complexType>
  <xs:key name="CMSV_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@CMSV_ID"/>
  </xs:key>
</xs:element>
```

Figure 160.Implementation of the CO_MDSV element

4.8.2. Element MOT – Media Object Type

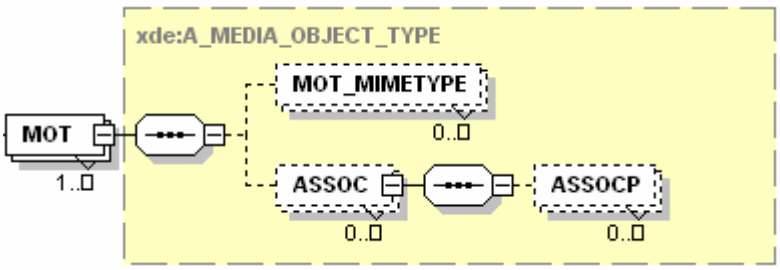


Figure 161.XML Schema of the MOT element

The `MOT` element corresponds to the `A_MEDIA_OBJECT_TYPES` table and represents `Media Object Type` and the set of its associations (Figure 161). The optional and unbounded `MOT_MIMETYPE` element points, by key reference `MTM_MT_ID`, to all related MIME-types. The `ASSOC` element defines possible associations with other Media Object Types by the key reference `ASS_MOT_ID`. The optional and unbounded child element `ASSOCP` represents the set of association parameters of the particular association.

Attributes:

Name	Type	Use
MOT_ID	xde:AT_ID	required
MOT_NAME	xde:AT_NAME	required
MOT_NATURE	xde:AT_MO_NATURE_TYPE	required
MOT_DESCRIPTION	xde:AT_DESCRIPTION	required
MOT_MDSS_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
key	MOT_ID		@MOT_ID
keyref	MOT_MDSS_ID	xde:MMS_ID	@MOT_MDSS_ID

Child element: `MOT_MIMETYPE`

Attributes:

Name	Type	Use
MTM_MT_ID	xs:decimal	required

Constraints:

	Name	Refer	Field(s)
keyref	MTM_MT_ID	xde:MT_ID	@MTM_MT_ID

Child element: ASSOC

Attributes:

Name	Type	Use
ASS_NAME	xde:AT_NAME	required
ASS_MOT_ID	xde:AT_ID	required
ASS_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	ASS_MOT_ID	xde:MOT_ID	@ ASS_MOT_ID
key	ASS_ID		@ ASS_ID

Child element: ASSOC/ASSOCP

Attributes:

Name	Type	Use
AP_NAME	xde:AT_NAME	required
AP_ID	xde:AT_ID	required
AP_DT_CODE	xde:AT_CODE	required

Constraints:

	Name	Refer	Field(s)
key	AP_ID		@AP_ID
keyref	AP_DT_CODE	xde:DT_CODE	@AP_DT_CODE

The implementation of the MOT element is shown in Figure 162, while its type definition in Figure 163.

```
<xs:element name="MOT" type="xde:A_MEDIA_OBJECT_TYPE" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_MEDIA_OBJ_TYPES"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="MOT_ID" seq="A_TYPES_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:key name="MOT_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@MOT_ID"/>
  </xs:key>
  <xs:keyref name="MOT_MDSS_ID" refer="xde:MMS_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@MOT_MDSS_ID"/>
  </xs:keyref>
</xs:element>
```

Figure 162.Implementation of the MOT element

```
<xs:complexType name="A_MEDIA_OBJECT_TYPE">
  <xs:annotation>
    <xs:appinfo>"map:A_MEDIA_OBJECT_TYPE=A_MEDIA_OBJECT_TYPES"</xs:appinfo>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="MOT_MIMETYPE" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Map name="A_MEDIA_OBJ_TYPES_TO_MIMETYPES"/>
            <Type value="exp"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="MTM_MOT_ID" parentAtt="MOT_ID"/>
            </Requirements>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="MTM_MT_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
      <xs:keyref name="MTM_MT_ID" refer="xde:MT_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@MTM_MT_ID"/>
      </xs:keyref>
    </xs:element>
    <xs:element name="ASSOC" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="exp"/>
            <Map name="A_ASSOCIATIONS"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="ASS_PARENT_MOT_ID" parentAtt="MOT_ID"/>
            </Requirements>
            <Sequencers>
              <Sequencer att="ASS_ID" seq="A_ASSOC_SEQ"/>
            </Sequencers>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:element name="ASSOCP" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="exp"/>
                  <Map name="A_ASSOC_PARAMETERS"/>
                  <ImportAction action="useOld"/>
                  <ErrorAction action="addNew"/>
                  <Requirements>
                    <Required att="AP_ASS_ID" parentAtt="ASS_ID"/>
                  </Requirements>
                  <Sequencers>
                    <Sequencer att="AP_ID" seq="A_PARAMS_SEQ"/>
                  </Sequencers>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="AP_NAME" type="xde:AT_NAME" use="required"/>
              <xs:attribute name="AP_ID" type="xde:AT_ID" use="required"/>
              <xs:attribute name="AP_DT_CODE" type="xde:AT_CODE" use="required"/>
```

```
            </xs:complexType>
            <xs:key name="AP_ID">
              <xs:selector xpath="."/>
              <xs:field xpath="@AP_ID"/>
            </xs:key>
            <xs:keyref name="AP_DT_CODE" refer="xde:DT_CODE">
              <xs:selector xpath="."/>
              <xs:field xpath="@AP_DT_CODE"/>
            </xs:keyref>
          </xs:element>
        </xs:sequence>
        <xs:attribute name="ASS_NAME" type="xde:AT_NAME" use="required"/>
        <xs:attribute name="ASS_MOT_ID" type="xde:AT_ID" use="required"/>
        <xs:attribute name="ASS_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
      <xs:keyref name="ASS_MOT_ID" refer="xde:MOT_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@ASS_MOT_ID"/>
      </xs:keyref>
      <xs:key name="ASS_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@ASS_ID"/>
      </xs:key>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="MOT_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="MOT_NAME" type="xde:AT_NAME" use="required"/>
  <xs:attribute name="MOT_NATURE" type="xde:AT_MO_NATURE_TYPE" use="required"/>
  <xs:attribute name="MOT_DESCRIPTION" type="xde:AT_DESCRIPTION" use="required"/>
  <xs:attribute name="MOT_MDSS_ID" type="xde:AT_ID" use="required"/>
</xs:complexType>
```

Figure 163.Implementation of the A_MEDIA_OBJECT_TYPE type

4.8.3. Element MO_MDS – MO Metadata Schema

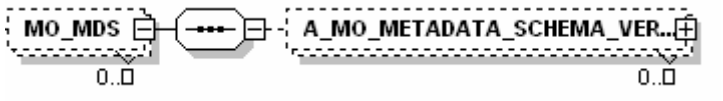


Figure 164.XML Schema of the MO_MDS element

The `MO_MDS` element corresponds to the `A_MO_METADATA_SCHEMA` table and represents Media Object AMS metadata schema (Figure 164). The `A_MO_METADATA_SCHEMA_VERSION` element represents specification of a Media Object schema version. The current Media Object schema version is selected by the key reference `MMS_CURRENT_MMSV_ID`.

Attributes:

Name	Type	Use
MMS_NAME	xde:AT_NAME	required
MMS_DESCRIPTION	xde:AT_DESCRIPTION	optional
MMS_ID	xde:AT_ID	required
MMS_CURRENT_MMSV_ID	xde:AT_ID	optional

Constraints:

	Name	Refer	Field(s)
key	MMS_ID		@MMS_ID
keyref	MMS_CURRENT_MMSV_ID	xde:MMSV_ID	@MMS_CURRENT_MMSV_ID

Child element: A_MO_METADATA_SCHEMA_VERSION

Attributes:

	Name	Refer	Field(s)
key	MMSV_ID		@MMSV_ID
keyref	MMSV_MDSS_ID	xde:MMS_ID	@MMSV_MDSS_ID

The implementation of the MO_MDS element is shown in Figure 165.

```
<xs:element name="MO_MDS" minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_MO_METADATA_SCHEMAS"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="MMS_ID" seq="A_SCHEMAS_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="A_MO_METADATA_SCHEMA_VERSION" minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
          <xs:appinfo>
            <XDM>
              <Type value="exp"/>
              <Map name="A_MO_METADATA_SCHEMA_VERSIONS"/>
              <ImportAction action="useOld"/>
              <ErrorAction action="addNew"/>
              <Requirements>
                <Required att="MMSV_MDSS_ID" parentAtt="MMS_ID"/>
              </Requirements>
              <Sequencers>
                <Sequencer att="MMSV_ID" seq="A_SCHEMAS_SEQ"/>
              </Sequencers>
            </XDM>
          </xs:appinfo>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="MMSV_SCHEMA" type="xde:AT_XML_AMS">
              <xs:annotation>
                <xs:appinfo>
                  <XDM>
                    <Type value="attEl"/>
                  </XDM>
                </xs:appinfo>
              </xs:annotation>
            </xs:element>
          </xs:sequence>
          <xs:attribute name="MMSV_ID" type="xde:AT_ID" use="required"/>
          <xs:attribute name="MMSV_VERSION" type="xde:AT_VERSION" use="required"/>
          <xs:attribute name="MMSV_ISSUED_ON" type="xde:AT_DATE" use="required"/>
        </xs:complexType>
        <xs:key name="MMSV_ID">
          <xs:selector xpath="."/>
          <xs:field xpath="@MMSV_ID"/>
        </xs:key>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="MMS_NAME" type="xde:AT_NAME" use="required"/>
    <xs:attribute name="MMS_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
    <xs:attribute name="MMS_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="MMS_CURRENT_MMSV_ID" type="xde:AT_ID" use="optional"/>
  </xs:complexType>
  <xs:key name="MMS_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@MMS_ID"/>
  </xs:key>
  <xs:keyref name="MMS_CURRENT_MMSV_ID" refer="xde:MMSV_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@MMS_CURRENT_MMSV_ID"/>
```

```
  </xs:keyref>
</xs:element>
```

Figure 165.Implementation of the MO_MDS element

4.8.4. Element MO_GEN_MDSV – General MO Metadata Schema Version

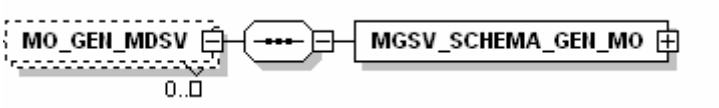


Figure 166.XML Schema of MO_GEN_MDSV element

The MO_GEN_MDSV element represents the A_MO_GEN_MD_SCHEMA_VERSIONS table and represents general Media Object AMS schema versions (Figure 166).

Constraints:

	Name	Refer	Field(s)
key	MGSV_ID		@MGSV_ID

The implementation of the MO_GEN_MDSV element is shown in Figure 167.

```
<xs:element name="MO_GEN_MDSV" minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_MO_GEN_MD_SCHEMA_VERSIONS"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="MGSV_ID" seq="A_SCHEMAS_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="MGSV_SCHEMA_GEN_MO" type="xde:AT_XML_AMS">
        <xs:annotation>
          <xs:appinfo>
            <XDM>
              <Type value="attEl"/>
            </XDM>
          </xs:appinfo>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="MGSV_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="MGSV_VERSION" type="xde:AT_VERSION" use="required"/>
    <xs:attribute name="MGSV_ISSUED_ON" type="xde:AT_DATE" use="required"/>
  </xs:complexType>
  <xs:key name="MGSV_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@MGSV_ID"/>
  </xs:key>
</xs:element>
```

Figure 167.Implementation of the MO_GEN_MDSV element

4.8.5. Element TD – Template Domains

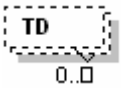


Figure 168.XML Schema of the TD element

The TD element corresponds to the A_TEMPLATE_DOMAINS table and represents the set of possible X-VRML Template domains (Figure 168).

Attributes:

Name	Type	Use
TD_ID	xde:AT_ID	required
TD_NAME	xde:AT_NAME	required
TD_DESCRIPTION	xde:AT_DESCRIPTION	optional

Constraints:

	Name	Refer	Field(s)
key	TD_ID		@TD_ID

The implementation of the TD element is shown in Figure 169.

```
<xs:element name="TD" minOccurs="0" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_TEMPLATE_DOMAINS"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="TD_ID" seq="A_TEMPLATES_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType name="A_TEMPLATE_DOMAINS">
    <xs:attribute name="TD_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="TD_NAME" type="xde:AT_NAME" use="required"/>
    <xs:attribute name="TD_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
  </xs:complexType>
  <xs:key name="TD_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TD_ID"/>
  </xs:key>
</xs:element>
```

Figure 169.Implementation of the TD element

4.8.6. Element XSL_TEMPLATE

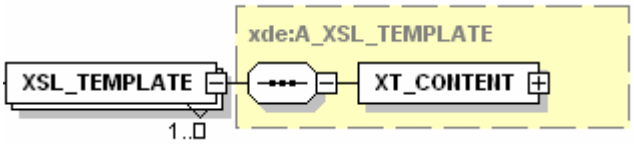


Figure 170.XML Schema of the XSL_TEMPLATE element

The XSL_TEMPLATE element represents an XSL template from the ARCO database.

Attributes:

Name	Type	Use
XT_ID	xde:AT_ID	required
XT_NAME	xde:AT_NAME	required
XT_DESCRIPTION	xde:AT_DESCRIPTION	optional

Constraints:

	Name	Refer	Field(s)
key	XT_ID		@XT_ID

The implementation of the XSL_TEMPLATE element is shown in Figure 171, while its type definition in Figure 172.

```
<xs:element name="XSL_TEMPLATE" type="xde:A_XSL_TEMPLATE" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_XSL_TEMPLATES"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="XT_ID" seq="A_TEMPLATES_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:key name="XT_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@XT_ID"/>
  </xs:key>
</xs:element>
```

Figure 171.Implementation of the XSL_TEMPLATE element

```
<xs:complexType name="A_XSL_TEMPLATE">
  <xs:sequence>
    <xs:element name="XT_CONTENT" type="xde:AT_XML_XSL">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="attEl"/>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="XT_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="XT_NAME" type="xde:AT_NAME" use="required"/>
  <xs:attribute name="XT_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
</xs:complexType>
```

Figure 172. Implementation of the A_XSL_TEMPLATE type

4.8.7. Element TOT – Template Object Type

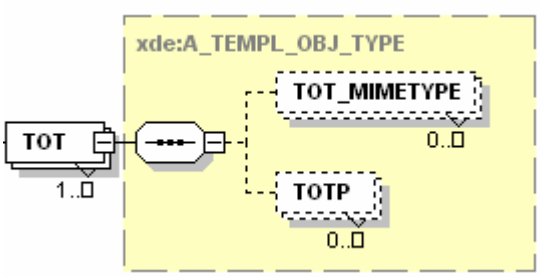


Figure 173. XML Schema of the TOT element

The `TOT` element corresponds to the `A_TEMPL_OBJ_TYPES` table and represents a Template Object Type (Figure 173). The optional and unbounded child element `TOT_MIMETYPE` element points, by key reference `TOMT_MT_ID`, to all allowed MIME-types. The optional and unbounded child element `TOTP` element defines the set of Template Object Type parameters.

Attributes:

Name	Type	Use
TOT_ID	xde:AT_ID	required
TOT_NAME	xde:AT_NAME	required
TOT_DESCRIPTION	xde:AT_DESCRIPTION	optional

Constraints:

	Name	Refer	Field(s)
key	TOT_ID		@TOT_ID

Child element: `TOT_MIMETYPE`

Attributes:

Name	Type	Use
TOMT_MT_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	TOMT_MT_ID	xde:MT_ID	@TOMT_MT_ID

Child element: TOTP

Attributes:

Name	Type	Use
TOTP_ID	xde:AT_ID	required
TOTP_NAME	xde:AT_NAME	required
TOTP_DESCRIPTION	xde:AT_DESCRIPTION	optional
TOTP_DT_CODE	xde:AT_CODE	optional

Constraints:

	Name	Refer	Field(s)
key	TOTP_ID		@TOTP_ID
keyref	TOTP_DT_CODE	xde:DT_CODE	@TOTP_DT_CODE

The implementation of the TOT element is shown in Figure 174, while its type definition in Figure 175.

```
<xs:element name="TOT" type="xde:A_TEMPL_OBJ_TYPE" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_TEMPL_OBJ_TYPES"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="TOT_ID" seq="A_TYPES_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:key name="TOT_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TOT_ID"/>
  </xs:key>
</xs:element>
```

Figure 174.Implementation of the TOT element

```
<xs:complexType name="A_TEMPL_OBJ_TYPE">
  <xs:sequence>
    <xs:element name="TOT_MIMETYPE" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="exp"/>
            <Map name="A_TEMPL_OBJ_TYPES_TO_MIMETYPES"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="TOMT_TOT_ID" parentAtt="TOT_ID"/>
            </Requirements>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="TOMT_MT_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
      <xs:keyref name="TOMT_MT_ID" refer="xde:MT_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@TOMT_MT_ID"/>
      </xs:keyref>
    </xs:element>
    <xs:element name="TOTP" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="exp"/>
            <Map name="A_TEMPL_OBJ_TYPE_PARAMS"/>
            <ImportAction action="useOld"/>
            <ErrorAction action="addNew"/>
            <Requirements>
              <Required att="TOTP_TOT_ID" parentAtt="TOT_ID"/>
            </Requirements>
            <Sequencers>
              <Sequencer att="TOTP_ID" seq="A_PARAMS_SEQ"/>
            </Sequencers>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="TOTP_ID" type="xde:AT_ID" use="required"/>
        <xs:attribute name="TOTP_NAME" type="xde:AT_NAME" use="required"/>
        <xs:attribute name="TOTP_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
        <xs:attribute name="TOTP_DT_CODE" type="xde:AT_CODE" use="optional"/>
      </xs:complexType>
      <xs:key name="TOTP_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@TOTP_ID"/>
      </xs:key>
      <xs:keyref name="TOTP_DT_CODE" refer="xde:DT_CODE">
        <xs:selector xpath="."/>
        <xs:field xpath="@TOTP_DT_CODE"/>
      </xs:keyref>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="TOT_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="TOT_NAME" type="xde:AT_NAME" use="required"/>
  <xs:attribute name="TOT_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
</xs:complexType>
```

Figure 175.Implementation of the A_TEMPL_OBJ_TYPE type

4.8.8. Element DATA_TYPE

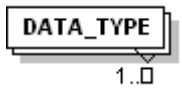


Figure 176.XML Schema of the DATA_TYPE element

The DATA_TYPE element represents a data type modelled in the ARCO database (Figure 176).

Attributes:

Name	Type	Use
DT_CODE	xde:AT_CODE	required
DT_NAME	xde:AT_NAME	required
DT_DESCRIPTION	xde:AT_DESCRIPTION	optional
DT_PATTERN	xde:AT_DATA_TYPE_PATTERN	optional
DT_VALUE_CATEGORY	xde:AT_DT_VALUE_CATEGORY	required

Constraints:

	Name	Refer	Field(s)
key	DT_CODE		@DT_CODE

The implementation of the A_DATA_TYPE element is shown in Figure 177.

```
<xs:element name="DATA_TYPE" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_DATA_TYPES"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType name="A_DATA_TYPES">
    <xs:attribute name="DT_CODE" type="xde:AT_CODE" use="required"/>
    <xs:attribute name="DT_NAME" type="xde:AT_NAME" use="required"/>
    <xs:attribute name="DT_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
    <xs:attribute name="DT_PATTERN" type="xde:AT_DATA_TYPE_PATTERN" use="optional"/>
    <xs:attribute name="DT_VALUE_CATEGORY" type="xde:AT_DT_VALUE_CATEGORY" use="required"/>
  </xs:complexType>
  <xs:key name="DT_CODE">
    <xs:selector xpath="."/>
    <xs:field xpath="@DT_CODE"/>
  </xs:key>
</xs:element>
```

Figure 177.Implementation of the DATA_TYPE element

4.8.9. Element MIME_TYPE

Figure 178.XML Schema of the MIME_TYPE element

The MIME_TYPE element represents a MIME-type definition used by ARCO contents (Figure 178).

Attributes:

Name	Type	Use
MT_ID	xde:AT_ID	required
MT_NAME	xde:AT_NAME	required
MT_EXTENSIONS	xde:AT_MIME_TYPE_EXTENSIONS	optional
MT_DESRIPTION	xde:AT_DESCRIPTION	optional

Constraints:

	Name	Refer	Field(s)
key	MT_ID		@MT_ID

The implementation of the A_MIME_TYPE element is shown in Figure 179.

```
<xs:element name="MIMETYPE" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_MIMETYPES"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="MT_ID" seq="A_DATA_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType name="A_MIME_TYPE">
    <xs:attribute name="MT_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="MT_NAME" type="xde:AT_NAME" use="required"/>
    <xs:attribute name="MT_EXTENSIONS" type="xde:AT_MIME_TYPE_EXTENSIONS" use="optional"/>
    <xs:attribute name="MT_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
  </xs:complexType>
  <xs:key name="MT_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@MT_ID"/>
  </xs:key>
</xs:element>
```

Figure 179.Implementation of the MIME_TYPE element

4.8.10. Element CF_DATA – Configuration Data

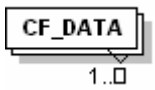


Figure 180.XML Schema of the CF_DATA element

The CF_DATA represents a value of an ARCO system configuration parameter (Figure 180).

Attributes:

Name	Type	Use
CO_ID	xde:AT_ID	required
CO_NAME	xde:AT_NAME	required

CO_VALUE	xde:AT_SHORT_STRING	optional

Constraints:

	Name	Refer	Field(s)
key	CO_ID_CONFIG		@CO_ID

The implementation of the CF_DATA element is shown in Figure 181.

```
<xs:element name="CONFIG_DATA" minOccurs="0">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="struct"/>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="CF_DATA" maxOccurs="unbounded">
        <xs:annotation>
          <xs:appinfo>
            <XDM>
              <Type value="exp"/>
              <Map name="A_CONFIG_DATA"/>
              <ImportAction action="useOld"/>
              <ErrorAction action="addNew"/>
              <Sequencers>
                <Sequencer att="CO_ID" seq="A_DATA_SEQ"/>
              </Sequencers>
            </XDM>
          </xs:appinfo>
        </xs:annotation>
        <xs:complexType name="A_CONFIG_DATA">
          <xs:attribute name="CO_ID" type="xde:AT_ID" use="required"/>
          <xs:attribute name="CO_NAME" type="xde:AT_NAME" use="required"/>
          <xs:attribute name="CO_VALUE" type="xde:AT_SHORT_STRING" use="optional"/>
        </xs:complexType>
        <xs:key name="CO_ID_CONFIG">
          <xs:selector xpath="."/>
          <xs:field xpath="@CO_ID"/>
        </xs:key>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Figure 181.Implementation of the CF_DATA element

4.8.11. Element PRO – Properties

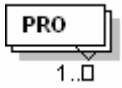


Figure 182.XML Schema of the PRO element

The PRO element represents a presentation property in the ARCO system (Figure 182). Presentation properties provide visualization parameters for ARIF Folders, Cultural Objects in ARIF Folders, and Media Objects in Cultural Objects in ARIF Folders.

Attributes:

Name	Type	Use
PRO_ID	xde:AT_ID	required
PRO_NAME	xde:AT_NAME	required
PRO_LEVEL	xde:AT_LEVEL	required
PRO_DT_CODE	xde:AT_CODE	optional

PRO_CATEGORY	xde:AT_PRO_CATEGORY	required
PRO_DESCRIPTION	xde:AT_DESCRIPTION	optional

Constraints:

	Name	Refer	Field(s)
key	PRO_ID		@PRO_ID
keyref	PRO_DT_CODE	xde:DT_CODE	@PRO_DT_CODE

The implementation of the PRO element is shown in Figure 183.

```
<xs:element name="PRO" maxOccurs="unbounded">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="exp"/>
        <Map name="A_PROPERTIES"/>
        <ImportAction action="useOld"/>
        <ErrorAction action="addNew"/>
        <Sequencers>
          <Sequencer att="PRO_ID" seq="A_OBJECTS_SEQ"/>
        </Sequencers>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType name="A_PROPERTY2">
    <xs:attribute name="PRO_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="PRO_NAME" type="xde:AT_NAME" use="required"/>
    <xs:attribute name="PRO_LEVEL" type="xde:AT_LEVEL" use="required"/>
    <xs:attribute name="PRO_DT_CODE" type="xde:AT_CODE" use="optional"/>
    <xs:attribute name="PRO_CATEGORY" type="xde:AT_PRO_CATEGORY" use="required"/>
    <xs:attribute name="PRO_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
  </xs:complexType>
  <xs:key name="PRO_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@PRO_ID"/>
  </xs:key>
  <xs:keyref name="PRO_DT_CODE" refer="xde:DT_CODE">
    <xs:selector xpath="."/>
    <xs:field xpath="@PRO_DT_CODE"/>
  </xs:keyref>
</xs:element>
```

Figure 183.Implementation of the PRO element

4.9. ARCO_DATA XML Schema Diagram

The XML Schema diagram of the XDE ARCO_DATA element is shown in Figure 184. The XML Schema listing of XDE is presented in Appendix N.

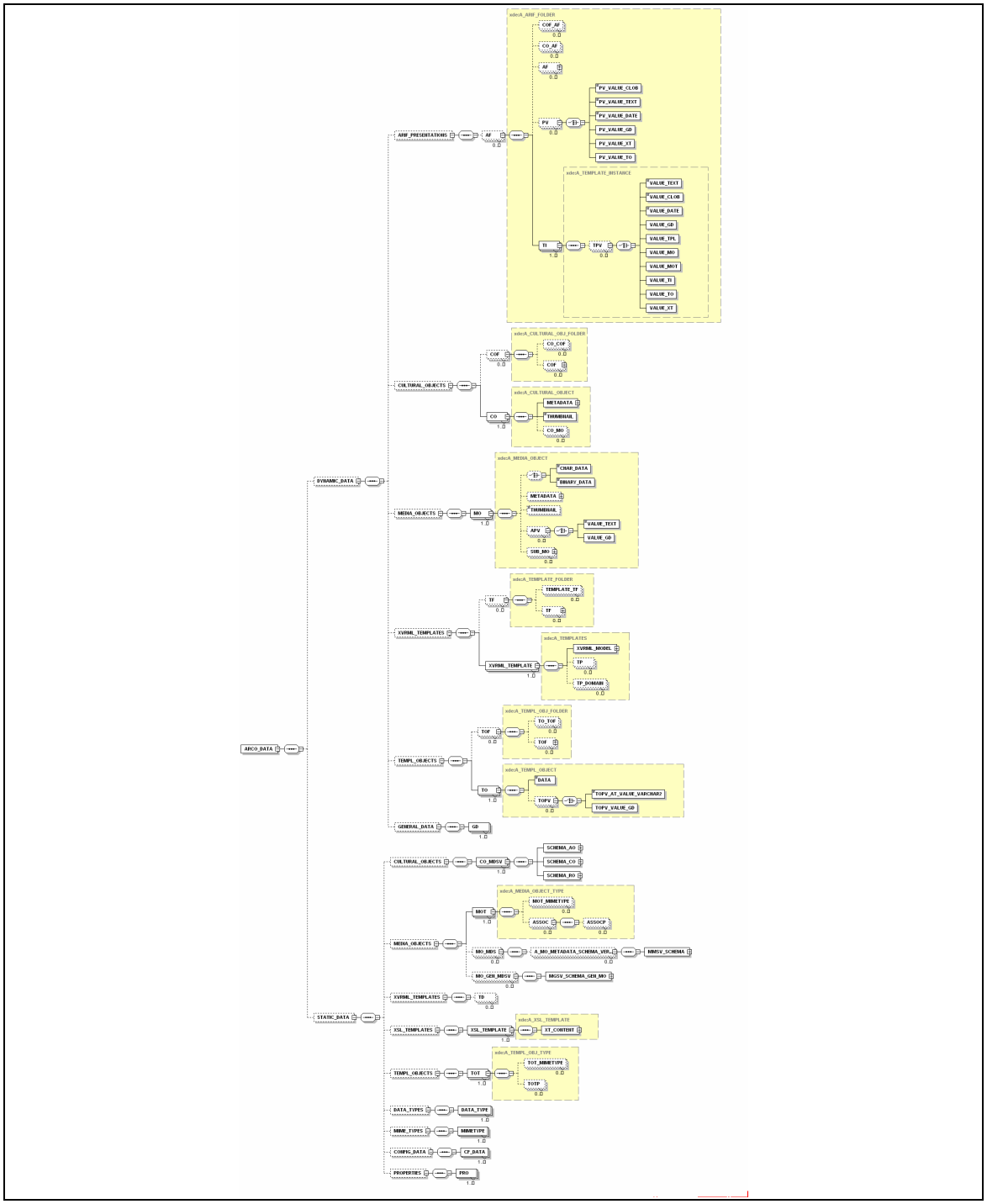


Figure 184.Structure of the XDE ARCO_DATA element

5. References

[1] ARCO Deliverable SFP-AMS – “Report on the XML descriptions of the database of cultural objects”, ARCO Consortium 2003

[2] ARCO Deliverable SFP-OM – "Object Modeller”, Specification of the Final ARCO Prototype, ARCO Consortium 2003

[3] ARCO Deliverable SFP-IMRR – “Interactive Model Refinement and Rendering tool”, Specification of the Final ARCO Prototype, ARCO Consortium 2003

[4] ARCO Deliverable SFP-ORDBMS – "Object Relational Database Management System", Specification of the Final ARCO Prototype, ARCO Consortium 2003

[5] ARCO Deliverable SFP-ARIF – "Augmented Reality Interface", Specification of the Final ARCO Prototype, ARCO Consortium 2003

[6] Database Working Group Charter, 1997; http://www.vrml.org/WorkingGroups/dbwork/charter.html

[7] VRML Recommended Practices for SQL Database Access, Request for Proposals, 1997; http://www.web3d.org/WorkingGroups/dbwork/dbrfp.html

[8] Lipkin, D., Database Extensions, Proposal for a VRML Informative Annex, Oracle Corporation, 1997; http://www.web3d.org/WorkingGroups/dbwork/oracledbex.html

[9] Lipkin, D., Recommended Practices for SQL Database Access - Background and Overview, Oracle Corporation, http://www.web3d.org/Recommended/vrml-sql/vrml-sql-ex.html

[10] Lipkin, D., Recommended Practices for SQL Database Access, VRML Informative Annex, 1998; http://www.web3d.org/Recommended/vrml-sql/

[11] 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

[12] Walczak K., Database Modeling of Virtual Reality, Doctoral Dissertation, Technical University of Gdansk, Poland, 2001

[13] 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

[14] 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

[15] Walczak, K., W. Cellary, X-VRML for Advanced Virtual Reality Applications, IEEE Computer, Volume 36, Nr 3; March 2003; IEEE Computer Society Press; pp. 89-92

[16] X-VRML website, http://xvrml.kti.ae.poznan.pl/

[17] XML - Extensible Markup Language, http://www.w3.org/XML/

[18] Java Servlet Technology, http://java.sun.com /products/servlet/

[19] Rachel Heery and Manjula Patel, “Application profiles: mixing and matching metadata schemas”, Ariadne, Issue 25, 2000, http://www.ariadne.ac.uk/issue25/app-profiles/

[20] A. Powell and P. Johnston, “Guidelines for implementing Dublin Core in XML”, http://www.dublincore.org/documents/dc-xml-guidelines/

[21] XML Schema Part 0: Primer Specification, http://www.w3.org/TR/xmlschema-0/

Appendix A. XML Schema for Administrative Metadata

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.arco-web.org/schemas/version11/administrative/"
xmlns:arco="http://www.arco-web.org/schemas/version11/application/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:adm="http://www.arco-
web.org/schemas/version11/administrative/" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <xsd:annotation>
      <xsd:documentation>
         <Project>ARCO</Project>
         <Protype>Final</Protype>
         <Date>02-09-03</Date>
         <Version>11</Version>
         <Creator>Nicholaos Mourkoussis email:N.Mourkoussis@sussex.ac.uk</Creator>
         <Contributor>Martin White email:m.white@sussex.ac.uk</Contributor>
         <Contributor>Manjula Patel email:m.patel@ukoln.ac.uk</Contributor>
         <Contributor>Jacek Chmielewski  email: jacek.chmielewski@kti.ae.poznan.pl</Contributor>
         <Contributor>Krzysztof Walczak email: walczak@kti.ae.poznan.pl</Contributor>
      </xsd:documentation>
   </xsd:annotation>
   <xsd:element name="administrativeMetadata">
      <xsd:annotation>
         <xsd:appinfo>
            <arco:displayName>Administrative Metadata</arco:displayName>
            <arco:definition>The cultural object Metadata Schema describes and structures
information concerning the creator of a metadata record and any modifications that are made to it.
Distinct instances are used to describe the modifications of all Cultural, Aquired, Refined and Media
object</arco:definition>
         </xsd:appinfo>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="creator" type="xsd:string">
               <xsd:annotation>
                  <xsd:appinfo>
                     <arco:displayName>Creator</arco:displayName>
                     <arco:identifier>http://purl.org/dc/elements/1.1/creator</arco:identifier>
                     <arco:autoGenerated>AdministrativeMedata_Creator</arco:autoGenerated>
                     <arco:example>e.g. Smith, John</arco:example>
                  </xsd:appinfo>
               </xsd:annotation>
            </xsd:element>
            <xsd:element name="created" type="xsd:dateTime">
               <xsd:annotation>
                  <xsd:appinfo>
                     <arco:displayName>Date Created</arco:displayName>
                     <arco:identifier>http://purl.org/dc/terms/created</arco:identifier>
                     <arco:autoGenerated>AdministrativeMedata_DateCreated</arco:autoGenerated>
                     <arco:example>e.g 1997-07-16T19:20:30+01:00</arco:example>
                  </xsd:appinfo>
               </xsd:annotation>
            </xsd:element>
            <xsd:element name="modified" type="xsd:dateTime">
               <xsd:annotation>
                  <xsd:appinfo>
                     <arco:displayName>Date Modified</arco:displayName>
                     <arco:identifier>http://www.dublincore.org/documents/dcmes-
qualifiers/modified </arco:identifier>
                     <arco:autoGenerated>AdministrativeMedata_DateModified</arco:autoGenerated>
                     <arco:example>1997-07-16T19:20:30+01:00</arco:example>
                  </xsd:appinfo>
               </xsd:annotation>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Appendix B. XML Schema for Cultural Object AMS

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.arco-web.org/schemas/version11/cultural/"
xmlns:clt="http://www.arco-web.org/schemas/version11/cultural/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:arco="http://www.arco-
web.org/schemas/version11/application/" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xsd:annotation>
        <xsd:documentation>
            <Project>ARCO</Project>
            <Protype>Final</Protype>
            <Date>02-09-03</Date>
            <Version>11</Version>
            <Creator>Nicholaos Mourkoussis email:N.Mourkoussis@sussex.ac.uk</Creator>
            <Contributor>Martin White email:m.white@sussex.ac.uk</Contributor>
            <Contributor>Manjula Patel email:m.patel@ukoln.ac.uk</Contributor>
            <Contributor>Jacek Chmielewski  email: jacek.chmielewski@kti.ae.poznan.pl</Contributor>
            <Contributor>Krzysztof Walczak email: walczak@kti.ae.poznan.pl</Contributor>
        </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" type="xsd:string">
                    <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 the museum).3 (a unique number for the donor).
5 (individual numbers for the objects within the donor's bequest)</arco:content>
                            <arco:example>LEWSA:1964.1.158</arco:example>
                        </xsd:appinfo>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="title" 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>This may be a common name or classification of an object in a
textual or codified form.</arco:content>
                            <arco:useType>Intelligence</arco:useType>
                            <arco:example>e.g. Castor || Mortar || Tray</arco:example>
                        </xsd:appinfo>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="alternative" 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>Can include name abbreviations as well as
translations.</arco:content>
                            <arco:useType>Intelligence</arco:useType>
                            <arco:example>e.g. Trophy || Blouse || Pram || Diary</arco:example>
                        </xsd:appinfo>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="creator" type="xsd:string" 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>
                            <arco:useType>Intelligence</arco:useType>
                            <arco:example>e.g. Smith, John || Morris Company || Phoenix
Ironworks</arco:example>
                        </xsd:appinfo>
                    </xsd:annotation>
                </xsd:element>
```

```
            <xsd:element name="contributor" type="xsd:string" 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 creator. 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>
                        <arco:useType>Intelligence</arco:useType>
                        <arco:example>e.g. Smith, John || Morris Company || Phoenix
Ironworks</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </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>This will correspond to the different types of physical object
represented as cultural objects in the database.</arco:content>
                        <arco:useType>Intelligence</arco:useType>
                        <arco:example>e.g. Coin || Vase || Pot || Ring</arco:example>
                    </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>
                        <arco:example>e.g. Shaped bow, probably once enamelled</arco:example>
                    </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 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>
                        <arco:example>e.g. All rights are held by the Victoria and Albert
Museum</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="objectProductionDate" type="xsd:string" minOccurs="0">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:displayName>Object Production Date</arco:displayName>
                        <arco:identifier>http://www.mda.org.uk/spectrum.htm/object-production-
date</arco:identifier>
                        <arco:definition>The date or period in which the cultural artefact was
created</arco:definition>
                        <arco:content>YYYYor MM.YYYYor DD.MM.YYYY or period</arco:content>
                        <arco:useType>Intelligence</arco:useType>
                        <arco:example>e.g. 1900 || 1900-10 || 1900-10-09 || roman</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="objectProductionPlace" type="xsd:string" minOccurs="0"
maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:displayName>Object Production Place</arco:displayName>
                        <arco:identifier>http://www.mda.org.uk/spectrum.htm/object-production-
place</arco:identifier>
                        <arco:definition>The geographical location in which an object was
created</arco:definition>
                        <arco:content>Record the name or title by which a place is normally known, or
a place reference number </arco:content>
                        <arco:useType>Intelligence</arco:useType>
                        <arco:example>e.g. Fishbourne, East Sussex, UK || FLG1992</arco:example>
                    </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.mda.org.uk/spectrum.htm/completeness</arco:identifier>
                        <arco:definition>The completeness of the object</arco:definition>
                        <arco:content>One of Complete, Incomplete, Uncertain</arco:content>
                        <arco:useType>Intelligence</arco:useType>
                        <arco:example>e.g. Complete </arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
                <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" type="xsd:string" minOccurs="0">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:displayName>Condition</arco:displayName>

   <arco:identifier>http://www.mda.org.uk/spectrum.htm/condition</arco:identifier>
                        <arco:definition>A single term describing the condition of an
object</arco:definition>
                        <arco:content>Record a single term or code.  The overall condition of the
object will be the same as the most serious individual condition which has been
identified</arco:content>
                        <arco:useType>Intelligence</arco:useType>
                        <arco:example>e.g. Poor || Fair || Good || 1 || 2 || A || B</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="productionPeriod" type="xsd:string" minOccurs="0">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:displayName>Production Period</arco:displayName>
                        <arco:identifier>http://www.mda.org.uk/spectrum.htm/production-
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, iIon Age
(using normal grammar and punctuation) or numerical form e.g., 1830=1860 (the = sign is always used
to separate components). This is used for cataloguing and is recorded only once</arco:content>
                        <arco:useType>Intelligence</arco:useType>
                        <arco:example>e.g. Late 19th century || Iron Age || medieval || Bronze Age ||
Ming</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="technique" type="xsd:string" minOccurs="0">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:displayName>Production Technique</arco:displayName>

   <arco:identifier>http://www.mda.org.uk/spectrum.htm/technique</arco:identifier>
                        <arco:definition>Processes, methods, techniques and tools used to fabricate
or decorate the object</arco:definition>
                        <arco:content>Use a simple term with no punctuation</arco:content>
                        <arco:useType>Intelligence</arco:useType>
                        <arco:example>e.g. Carved || Painted || Turned || Pencil</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="material" type="xsd:string" maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:displayName>Material</arco:displayName>

   <arco:identifier>http://www.mda.org.uk/spectrum.htm/material</arco:identifier>
                        <arco:definition>The basic material from which an object is
constructed</arco:definition>
                        <arco:content>Use a single term, without punctuation.  Do not include brand
names.</arco:content>
                        <arco:useType>Intelligence</arco:useType>
                        <arco:example>e.g. Clay || Amber || Tin || Brass || Wood || Steel Metal
(bell)</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="dimension" minOccurs="0" maxOccurs="unbounded">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:definition>A complex element which is used so as to group all the
dimension related elements</arco:definition>
                        <arco:content>Not Editable</arco:content>
                    </xsd:appinfo>
                </xsd:annotation>
                <xsd:complexType>
                    <xsd:sequence>
```

```
                        <xsd:element name="dimensionMeasuredPart" type="xsd:string" minOccurs="0">
                            <xsd:annotation>
                                <xsd:appinfo>
                                    <arco:displayName>Dimension Measured Part</arco:displayName>
                                    <arco:identifier>http://www.mda.org.uk/spectrum.htm/dimension-
measured-part</arco:identifier>
                                    <arco:definition>The part of the object that has been
measured</arco:definition>
                                    <arco:content>Use a single term to describe the part of the object
for which the dimension was measured</arco:content>
                                    <arco:example>e.g. Base  ||  Frame || Mount</arco:example>
                                </xsd:appinfo>
                            </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="dimension" type="xsd:string">
                            <xsd:annotation>
                                <xsd:appinfo>
                                    <arco:displayName>Dimension</arco:displayName>

   <arco:identifier>http://www.mda.org.uk/spectrum.htm/dimension</arco:identifier>
                                    <arco:definition>The aspect of a part or component of an object
being measured</arco:definition>
                                    <arco:content>Record a single term describing the dimension to be
measured</arco:content>
                                    <arco:example>e.g. Height || Width || Depth || Diameter || Weight
|| Circumference || Radius</arco:example>
                                </xsd:appinfo>
                            </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="dimensionValue" type="xsd:string">
                            <xsd:annotation>
                                <xsd:appinfo>
                                    <arco:displayName>Dimension Value</arco:displayName>
                                    <arco:identifier>http://www.mda.org.uk/spectrum.htm/dimension-
value</arco:identifier>
                                    <arco:definition>The numeric value of the measurement of a
dimension</arco:definition>
                                    <arco:content>Record a number without punctuation other than a
decimal point when required.  The value should be to the nearest point of detail dictated by the type
of object and the needs of the procedure </arco:content>
                                    <arco:example>e.g. 987  ||  0.234</arco:example>
                                </xsd:appinfo>
                            </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="dimensionMeasurementUnit" type="xsd:string">
                            <xsd:annotation>
                                <xsd:appinfo>
                                    <arco:displayName>Dimension Measurement Unit</arco:displayName>
                                    <arco:identifier>http://www.mda.org.uk/spectrum.htm/dimension-
measurement-value</arco:identifier>
                                    <arco:definition>The unit of measurement used when measuring a
dimension</arco:definition>
                                    <arco:content>Record a single term or abbreviation</arco:content>
                                    <arco:example>e.g. m || mm || inches || grains ||
kgs</arco:example>
                                </xsd:appinfo>
                            </xsd:annotation>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="components" minOccurs="0">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:displayName>Components</arco:displayName>
                        <arco:identifier>http://www.arco-
web.org/schemas/arco/components</arco:identifier>
                        <arco:definition>The description of any parts/pieces which comprise the
cultural artefact</arco:definition>
                        <arco:content>Record one object followed by a list of child
components</arco:content>
                        <arco:example>e.g. Tea Set: Tea cup, spoon, saucer, teapot, sugar
bowl</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="component" type="xsd:string" maxOccurs="unbounded"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
            <xsd:element name="acquisitionSource" type="xsd:string" minOccurs="0">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:displayName>Acquisition Source</arco:displayName>
                        <arco:identifier>http://www.mda.org.uk/spectrum.htm/acquisition-
source</arco:identifier>
                        <arco:definition>The organisation or person, or people from whom the object
was obtained.  For archaeological archives use to record the excavating body responsible for
preparing and depositing the object with the museum.</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>
```

```
                        <arco:useType>Intelligence</arco:useType>
                        <arco:example/>
                        <arco:example>e.g. Smith, John</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="accessionDate" type="xsd:string">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:displayName>Accession Date</arco:displayName>
                        <arco:identifier>http://www.mda.org.uk/spectrum.htm/accession-
date</arco:identifier>
                        <arco:definition>The date on which an object formally enters the collections
and is recorded into the accessions register</arco:definition>
                        <arco:content>Always record the day, month and year in the same order, use
the following format: DDMMYYYY
</arco:content>
                        <arco:useType>Intelligence</arco:useType>
                        <arco:example>e.g. 08101993</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="currentLocation" type="xsd:string">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:displayName>Current Location</arco:displayName>
                        <arco:identifier>http://www.mda.org.uk/spectrum.htm/current-
location</arco:identifier>
                        <arco:definition>The physical place within an institution where an object is
currently located</arco:definition>
                        <arco:content>Record a term or code describing the current whereabouts of an
object according to standard practice in the institution.</arco:content>
                        <arco:example>e.g. U.23.2.4a room 1, case 3, shelf 4</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="fieldCollection">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:definition>A complex element which is used so as to group all the field
collection related elements</arco:definition>
                        <arco:content>Not Editable</arco:content>
                    </xsd:appinfo>
                </xsd:annotation>
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="fieldCollectionDate" type="xsd:string">
                            <xsd:annotation>
                                <xsd:appinfo>
                                    <arco:displayName>Field Collection Date</arco:displayName>
                                    <arco:identifier>http://www.mda.org.uk/spectrum.htm/field-
collection-date</arco:identifier>
                                    <arco:definition>The date an object was collected in the
field</arco:definition>
                                    <arco:content>Always record the day, month and year in the same
order, use the following format: DDMMYYYY. It may be necessary to record a range</arco:content>
                                    <arco:example>e.g. 09031965</arco:example>
                                </xsd:appinfo>
                            </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="fieldCollectionMethod" type="xsd:string" minOccurs="0">
                            <xsd:annotation>
                                <xsd:appinfo>
                                    <arco:displayName>Field Collection Method</arco:displayName>
                                    <arco:identifier>http://www.mda.org.uk/spectrum.htm/field-
collection-method</arco:identifier>
                                    <arco:definition>The method used to excavate or collect an object
in the field</arco:definition>
                                    <arco:content>Use a single term without punctuation</arco:content>
                                    <arco:example>e.g. Netted  || shot || trapped || borehole ||
photographed</arco:example>
                                </xsd:appinfo>
                            </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="fieldCollectionPlaceName" type="xsd:string" minOccurs="0">
                            <xsd:annotation>
                                <xsd:appinfo>
                                    <arco:displayName>Field Collection Place</arco:displayName>
                                    <arco:identifier>http://www.mda.org.uk/spectrum.htm/field-
collection-place-name</arco:identifier>
                                    <arco:definition>The name of the place where an object was
collected in the field</arco:definition>
                                    <arco:content>The name or title by which the place is normally
known</arco:content>
                                    <arco:example>e.g. Australia ||  Orkney || Manhattan</arco:example>
                                </xsd:appinfo>
                            </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="fieldCollector" type="xsd:string" minOccurs="0">
                            <xsd:annotation>
                                <xsd:appinfo>
                                    <arco:displayName>Field Collector</arco:displayName>
```

```
                                    <arco:identifier>http://www.mda.org.uk/spectrum.htm/field-
collector</arco:identifier>
                                    <arco:definition>The person or organisation responsible for
collecting a specimen or object in the field</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>
                                    <arco:example>e.g. VAM, Photographic Unit || Sussex Archeological
Society </arco:example>
                                </xsd:appinfo>
                            </xsd:annotation>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:complexType>
            </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 targetNamespace="http://www.arco-web.org/schemas/version11/acquired/"
xmlns:acq="http://www.arco-web.org/schemas/version11/acquired/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:arco="http://www.arco-
web.org/schemas/version11/application/" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xsd:annotation>
        <xsd:documentation>
            <Project>ARCO</Project>
            <Protype>Final</Protype>
            <Date>02-09-03</Date>
            <Version>11</Version>
            <Creator>Nicholaos Mourkoussis email:N.Mourkoussis@sussex.ac.uk</Creator>
            <Contributor>Martin White email:m.white@sussex.ac.uk</Contributor>
            <Contributor>Manjula Patel email:m.patel@ukoln.ac.uk</Contributor>
            <Contributor>Jacek Chmielewski  email: jacek.chmielewski@kti.ae.poznan.pl</Contributor>
            <Contributor>Krzysztof Walczak email: walczak@kti.ae.poznan.pl</Contributor>
        </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" type="xsd:string">
                    <xsd:annotation>
                        <xsd:appinfo>
                            <arco:autoGenerated>AO_Identifier</arco:autoGenerated>
                            <arco:displayName>Identifier</arco:displayName>
                            <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>
                            <arco:example>e.g. PAST000004005</arco:example>
                        </xsd:appinfo>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="title" 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>
                            <arco:example>e.g. Vase without hole</arco:example>
                        </xsd:appinfo>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="publisher" type="xsd:string" 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>
                            <arco:example>e.g. ARCO Project Consortium || VAM, Photographic Unit ||
Smith, John</arco:example>
                        </xsd:appinfo>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="creator" type="xsd:string">
                    <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>
                            <arco:example>e.g. Smith, John</arco:example>
                        </xsd:appinfo>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="contributor" type="xsd:string" 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>
                        <arco:example>e.g. Victoria and Albert Museum, Photographic
Unit</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="created" 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>
                        <arco:example>e.g. 1997-07-16T19:20:30+01:00</arco:example>
                    </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 Acquired
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>
                        <arco:example>e.g. http://www.nhm.ac.uk/generic/copy.html</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="format" type="xsd:string">
                <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 acquired Object</arco:definition>
                        <arco:content>Not Editable</arco:content>
                        <arco:example>e.g. text/html</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="extent" type="xsd:string">
                <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 acquired Object</arco:definition>
                        <arco:content>Not Editable</arco:content>
                        <arco:example>e.g. 12 KB </arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </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 targetNamespace="http://www.arco-web.org/schemas/version11/refined/"
xmlns:rfn="http://www.arco-web.org/schemas/version11/refined/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:arco="http://www.arco-
web.org/schemas/version11/application/" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xsd:annotation>
        <xsd:documentation>
            <Project>ARCO</Project>
            <Protype>Final</Protype>
            <Date>02-09-03</Date>
            <Version>11</Version>
            <Creator>Nicholaos Mourkoussis email:N.Mourkoussis@sussex.ac.uk</Creator>
            <Contributor>Martin White email:m.white@sussex.ac.uk</Contributor>
            <Contributor>Manjula Patel email:m.patel@ukoln.ac.uk</Contributor>
            <Contributor>Jacek Chmielewski  email: jacek.chmielewski@kti.ae.poznan.pl</Contributor>
            <Contributor>Krzysztof Walczak email: walczak@kti.ae.poznan.pl</Contributor>
        </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" type="xsd:string">
                    <xsd:annotation>
                        <xsd:appinfo>
                            <arco:autoGenerated>AO_Identifier</arco:autoGenerated>
                            <arco:displayName>Identifier</arco:displayName>
                            <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>
                            <arco:example>e.g. PAST000004005</arco:example>
                        </xsd:appinfo>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="title" 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 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>
                            <arco:example>e.g. Vase without hole</arco:example>
                        </xsd:appinfo>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="publisher" type="xsd:string" 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>
                            <arco:example>e.g. ARCO Project Consortium || VAM, Photographic Unit ||
Smith, John</arco:example>
                        </xsd:appinfo>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="creator" type="xsd:string">
                    <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>
                            <arco:example>e.g. Smith, John</arco:example>
                        </xsd:appinfo>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="contributor" type="xsd:string" 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>
                        <arco:example>e.g. Victoria and Albert Museum, Photographic
Unit</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="created" 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 refined Object</arco:definition>
                        <arco:content>Not Editable</arco:content>
                        <arco:example>e.g. 1997-07-16T19:20:30+01:00</arco:example>
                    </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 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 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>
                        <arco:example>e.g. http://www.nhm.ac.uk/generic/copy.html</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="format" type="xsd:string">
                <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>
                        <arco:example>e.g. text/html</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="extent" type="xsd:string">
                <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>
                        <arco:example>e.g. 12 KB </arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="isVersionOf" type="xsd:string">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:autoGenerated>RO_RelationIsVersionOf</arco:autoGenerated>
                        <arco:displayName>Relation Is Version Of</arco:displayName>
                        <arco:identifier>http://pur.org/dc/terms/isVersionOf</arco:identifier>
                        <arco:definition>Can be either an AO-Identifier or an RO-
Identifier</arco:definition>
                        <arco:content>Not Editable</arco:content>
                        <arco:example>e.g. VAM00000123, PAST00000124</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </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 targetNamespace="http://www.arco-web.org/schemas/version11/media/"
xmlns:arco="http://www.arco-web.org/schemas/version11/application/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:mdo="http://www.arco-
web.org/schemas/version11/media/" elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:annotation>
      <xsd:documentation>
         <Project>ARCO</Project>
         <Protype>Final</Protype>
         <Date>02-09-03</Date>
         <Version>11</Version>
         <Creator>Nicholaos Mourkoussis email:N.Mourkoussis@sussex.ac.uk</Creator>
         <Contributor>Martin White email:m.white@sussex.ac.uk</Contributor>
         <Contributor>Manjula Patel email:m.patel@ukoln.ac.uk</Contributor>
         <Contributor>Jacek Chmielewski  email: jacek.chmielewski@kti.ae.poznan.pl</Contributor>
         <Contributor>Krzysztof Walczak email: walczak@kti.ae.poznan.pl</Contributor>
      </xsd:documentation>
   </xsd:annotation>
   <xsd:element name="mediaObject">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="title" 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>
                     <arco:example>e.g. My vase</arco:example>
                  </xsd:appinfo>
               </xsd:annotation>
            </xsd:element>
            <xsd:element name="type" type="xsd:string">
               <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>
                     <arco:example>e.g. model/vrml</arco:example>
                  </xsd:appinfo>
               </xsd:annotation>
            </xsd:element>
            <xsd:element name="subject" type="xsd:string" 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>Use capitals for the first letter in each word and delimit by
comma.</arco:content>
                     <arco:example>e.g. food  || coronation || garden party ||
elephant</arco:example>
                  </xsd:appinfo>
               </xsd:annotation>
            </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>
                     <arco:useType>Effort</arco:useType>
                     <arco:example>e.g. The coronation of Queen Elizabeth II</arco:example>
                  </xsd:appinfo>
               </xsd:annotation>
            </xsd:element>
            <xsd:element name="created" 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>
                     <arco:example>e.g. 1997-07-16T19:20:30+01:00</arco:example>
                  </xsd:appinfo>
```

```
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="technique" type="xsd:string" 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>It should be a simple keyword</arco:content>
                        <arco:useType>Effort</arco:useType>
                        <arco:example>e.g. Digital photography, Scanning</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="creator" type="xsd:string">
                <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>
                        <arco:useType>Effort</arco:useType>
                        <arco:example>e.g. Smith, John</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="extent" type="xsd:string">
                <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>
                        <arco:example>e.g. 1 KB || 3 MB</arco:example>
                    </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 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>
                        <arco:example>e.g. All rights are held by the Victoria and Albert
Museum</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="skillLevel" type="xsd:string">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:displayName>Skill Level</arco:displayName>
                        <arco:identifier>http://www.arco-web.org/schemas/arco/skill-
level</arco:identifier>
                        <arco:definition>The skill level or experience of the creator of the Media
Object</arco:definition>
                        <arco:content>Describe using terms such as novice, experienced, very
experienced</arco:content>
                        <arco:useType>Effort</arco:useType>
                        <arco:example>e.g. Novice || Experienced || Very Experienced</arco:example>
                    </xsd:appinfo>
                </xsd:annotation>
            </xsd:element>
            <xsd:element name="personEffort" type="xsd:decimal">
                <xsd:annotation>
                    <xsd:appinfo>
                        <arco:displayName>Person Effort</arco:displayName>
                        <arco:identifier>http://www.arco-web.org/schemas/arco/person-
effort</arco:identifier>
                        <arco:definition>The amount of time spent in capturing the Media
Object</arco:definition>
                        <arco:content>Enter the amount of time in hours</arco:content>
                        <arco:useType>Effort</arco:useType>
                        <arco:example>e.g. 3.5 || 9.3 || 7.0</arco:example>
                    </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 targetNamespace="http://www.arco-web.org/schemas/version11/panoramic/"
xmlns:mpi="http://www.arco-web.org/schemas/version11/panoramic/" xmlns:arco="http://www.arco-
web.org/schemas/version11/application/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:annotation>
      <xsd:documentation>
         <Project>ARCO</Project>
         <Protype>Final</Protype>
         <Date>02-09-03</Date>
         <Version>11</Version>
         <Creator>Nicholaos Mourkoussis email:N.Mourkoussis@sussex.ac.uk</Creator>
         <Contributor>Martin White email:m.white@sussex.ac.uk</Contributor>
         <Contributor>Manjula Patel email:m.patel@ukoln.ac.uk</Contributor>
         <Contributor>Jacek Chmielewski  email: jacek.chmielewski@kti.ae.poznan.pl</Contributor>
         <Contributor>Krzysztof Walczak email: walczak@kti.ae.poznan.pl</Contributor>
      </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="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>
                     <arco:example>e.g. 19 || 25</arco:example>
                  </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>
                     <arco:example>e.g. 3.0 || 20.0</arco:example>
                  </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 targetNamespace="http://www.arco-web.org/schemas/version11/description/"
xmlns:mde="http://www.arco-web.org/schemas/version11/description/" xmlns:arco="http://www.arco-
web.org/schemas/version11/application/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:annotation>
      <xsd:documentation>
         <Project>ARCO</Project>
         <Protype>Final</Protype>
         <Date>02-09-03</Date>
         <Version>11</Version>
         <Creator>Nicholaos Mourkoussis email:N.Mourkoussis@sussex.ac.uk</Creator>
         <Contributor>Martin White email:m.white@sussex.ac.uk</Contributor>
         <Contributor>Manjula Patel email:m.patel@ukoln.ac.uk</Contributor>
         <Contributor>Jacek Chmielewski  email: jacek.chmielewski@kti.ae.poznan.pl</Contributor>
         <Contributor>Krzysztof Walczak email: walczak@kti.ae.poznan.pl</Contributor>
      </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="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>
                     <arco:example>e.g. 2596 || 1212</arco:example>
                  </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>
                     <arco:example>e.g. ISO-8859-2</arco:example>
                  </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: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-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>
        </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 targetNamespace="http://www.arco-web.org/schemas/version11/max/"
xmlns:arco="http://www.arco-web.org/schemas/version11/application/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:mmx="http://www.arco-
web.org/schemas/version11/max/" elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:annotation>
      <xsd:documentation>
         <Project>ARCO</Project>
         <Protype>Final</Protype>
         <Date>02-09-03</Date>
         <Version>11</Version>
         <Creator>Nicholaos Mourkoussis email:N.Mourkoussis@sussex.ac.uk</Creator>
         <Contributor>Martin White email:m.white@sussex.ac.uk</Contributor>
         <Contributor>Manjula Patel email:m.patel@ukoln.ac.uk</Contributor>
         <Contributor>Jacek Chmielewski  email: jacek.chmielewski@kti.ae.poznan.pl</Contributor>
         <Contributor>Krzysztof Walczak email: walczak@kti.ae.poznan.pl</Contributor>
      </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="softwareVersion" type="xsd:string">
               <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>A numeric value. Also it should be able to accept any future
version of the software.</arco:content>
                     <arco:example>e.g. 3.0 || 4.0 || 4.2 || 5.0</arco:example>
                  </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>
                     <arco:example>e.g. Brick, BlurFire, ARCO plug-in</arco:example>
                  </xsd:appinfo>
               </xsd:annotation>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Appendix I. XML Schema for Media Object VRML Model AMS

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.arco-web.org/schemas/version11/vrml/"
xmlns:arco="http://www.arco-web.org/schemas/version11/application/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:vrm="http://www.arco-
web.org/schemas/version11/vrml/" elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:annotation>
      <xsd:documentation>
         <Project>ARCO</Project>
         <Protype>Final</Protype>
         <Date>02-09-03</Date>
         <Version>11</Version>
         <Creator>Nicholaos Mourkoussis email:N.Mourkoussis@sussex.ac.uk</Creator>
         <Contributor>Martin White email:m.white@sussex.ac.uk</Contributor>
         <Contributor>Manjula Patel email:m.patel@ukoln.ac.uk</Contributor>
         <Contributor>Jacek Chmielewski  email: jacek.chmielewski@kti.ae.poznan.pl</Contributor>
         <Contributor>Krzysztof Walczak email: walczak@kti.ae.poznan.pl</Contributor>
      </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="vrmlVersion" type="xsd:string">
               <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>
                     <arco:example>e.g. 1.0 || 2.0 || 97 || 200x || X3D</arco:example>
                  </xsd:appinfo>
               </xsd:annotation>
            </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>
                     <arco:example>e.g. 147 || 15 || 8</arco:example>
                  </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>
                     <arco:example>e.g. TRUE  || FALSE</arco:example>
                  </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>
                     <arco:example>e.g. TRUE || FALSE</arco:example>
                  </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 targetNamespace="http://www.arco-web.org/schemas/version11/panoramic/"
xmlns:mpi="http://www.arco-web.org/schemas/version11/panoramic/" xmlns:arco="http://www.arco-
web.org/schemas/version11/application/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
   <xsd:annotation>
      <xsd:documentation>
         <Project>ARCO</Project>
         <Protype>Final</Protype>
         <Date>02-09-03</Date>
         <Version>11</Version>
         <Creator>Nicholaos Mourkoussis email:N.Mourkoussis@sussex.ac.uk</Creator>
         <Contributor>Martin White email:m.white@sussex.ac.uk</Contributor>
         <Contributor>Manjula Patel email:m.patel@ukoln.ac.uk</Contributor>
         <Contributor>Jacek Chmielewski  email: jacek.chmielewski@kti.ae.poznan.pl</Contributor>
         <Contributor>Krzysztof Walczak email: walczak@kti.ae.poznan.pl</Contributor>
      </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="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>
                     <arco:example>e.g. 19 || 25</arco:example>
                  </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>
                     <arco:example>e.g. 3.0 || 20.0</arco:example>
                  </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 targetNamespace="http://www.arco-web.org/schemas/version11/multiresolution"
xmlns:arco="http://www.arco-web.org/schemas/version11/application/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:mmi="http://www.arco-
web.org/schemas/version11/multiresolution" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xsd:annotation>
        <xsd:documentation>
            <Project>ARCO</Project>
            <Protype>Final</Protype>
            <Date>02-09-03</Date>
            <Version>11</Version>
            <Creator>Nicholaos Mourkoussis email:N.Mourkoussis@sussex.ac.uk</Creator>
            <Contributor>Martin White email:m.white@sussex.ac.uk</Contributor>
            <Contributor>Manjula Patel email:m.patel@ukoln.ac.uk</Contributor>
            <Contributor>Jacek Chmielewski  email: jacek.chmielewski@kti.ae.poznan.pl</Contributor>
            <Contributor>Krzysztof Walczak email: walczak@kti.ae.poznan.pl</Contributor>
        </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="resolutions" type="xsd:string">
                    <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>
                            <arco:example>e.g. 640x480, 800x600, 1024x768</arco:example>
                        </xsd:appinfo>
                    </xsd:annotation>
                </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>
                            <arco:example>e.g. Adobe Photoshop</arco:example>
                        </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>
                            <arco:example>e.g. Area averaging</arco:example>
                        </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: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>
```

Appendix N. Implementation of the XDE in the Final ARCO Prototype

```
<xs:element name="ARCO_DATA">
  <xs:annotation>
    <xs:appinfo>
      <XDM>
        <Type value="struct"/>
      </XDM>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="DYNAMIC_DATA" minOccurs="0">
        <xs:annotation>
          <xs:appinfo>
            <XDM>
              <Type value="struct"/>
            </XDM>
          </xs:appinfo>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="ARIF_PRESENTATIONS" minOccurs="0">
              <xs:annotation>
                <xs:appinfo>
                  <XDM>
                    <Type value="struct"/>
                  </XDM>
                </xs:appinfo>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="AF" type="xde:A_ARIF_FOLDER" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                      <xs:appinfo>
                        <XDM>
                          <Type value="exp"/>
                          <Map name="A_ARIF_FOLDERS"/>
                          <ImportAction action="useOld"/>
                          <ErrorAction action="addNew"/>
                          <Sequencers>
                            <Sequencer att="AF_ID" seq="A_FOLDERS_SEQ"/>
                          </Sequencers>
                          <Requirements>
                            <Required att="AF_PARENT_FOLD_ID" parentAtt="AF_ID"/>
                          </Requirements>
                        </XDM>
                      </xs:appinfo>
                    </xs:annotation>
                    <xs:key name="AF_ID">
                      <xs:selector xpath="."/>
                      <xs:field xpath="@AF_ID"/>
                    </xs:key>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="CULTURAL_OBJECTS" minOccurs="0">
              <xs:annotation>
                <xs:appinfo>
                  <XDM>
                    <Type value="struct"/>
                  </XDM>
                </xs:appinfo>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="COF" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                      <xs:appinfo>
                        <XDM>
                          <Type value="exp"/>
                          <Map name="A_CULTURAL_OBJ_FOLDERS"/>
                          <ImportAction action="addNew">
                            <ImportAlter att="COF_NAME"/>
                          </ImportAction>
                          <ErrorAction action="addNew">
                            <ErrorAlter att="COF_NAME"/>
                          </ErrorAction>
                          <IdentifyingAttributes>
                            <Attribute name="COF_NAME"/>
                          </IdentifyingAttributes>
                          <Sequencers>
                            <Sequencer att="COF_ID" seq="A_FOLDERS_SEQ"/>
                          </Sequencers>
                          <Requirements>
                            <Required att="COF_PARENT_FOLDER_ID" parentAtt="COF_ID"/>
                          </Requirements>
```

```
                    </XDM>
                  </xs:appinfo>
                </xs:annotation>
                <xs:complexType>
                  <xs:complexContent>
                    <xs:extension base="xde:A_CULTURAL_OBJ_FOLDER"/>
                  </xs:complexContent>
                </xs:complexType>
                <xs:key name="COF_ID">
                  <xs:selector xpath="."/>
                  <xs:field xpath="@COF_ID"/>
                </xs:key>
                <xs:keyref name="COF_PARENT_FOLDER_ID" refer="xde:COF_ID">
                  <xs:selector xpath="."/>
                  <xs:field xpath="@COF_PARENT_FOLDER_ID"/>
                </xs:keyref>
              </xs:element>
              <xs:element name="CO" maxOccurs="unbounded">
                <xs:annotation>
                  <xs:appinfo>
                    <XDM>
                      <Type value="exp"/>
                      <Map name="A_CULTURAL_OBJECTS"/>
                      <ImportAction action="addNew"/>
                      <ErrorAction action="addNew">
                        <ErrorAlter att="CO_NAME"/>
                      </ErrorAction>
                      <IdentifyingAttributes>
                        <Attribute name="CO_NAME"/>
                      </IdentifyingAttributes>
                      <Sequencers>
                        <Sequencer att="CO_ID" seq="A_OBJECTS_SEQ"/>
                      </Sequencers>
                      <Requirements>
                        <Required att="CO_CREATED_FROM_CO_ID" parentAtt="CO_ID"/>
                      </Requirements>
                    </XDM>
                  </xs:appinfo>
                </xs:annotation>
                <xs:complexType>
                  <xs:complexContent>
                    <xs:extension base="xde:A_CULTURAL_OBJECT"/>
                  </xs:complexContent>
                </xs:complexType>
                <xs:key name="CO_ID">
                  <xs:selector xpath="."/>
                  <xs:field xpath="@CO_ID"/>
                </xs:key>
                <xs:keyref name="CO_CMSV_ID" refer="xde:CMSV_ID">
                  <xs:selector xpath="."/>
                  <xs:field xpath="@CO_CMSV_ID"/>
                </xs:keyref>
                <xs:keyref name="CO_CREATED_FROM_CO_ID" refer="xde:CO_ID">
                  <xs:selector xpath="."/>
                  <xs:field xpath="@CO_CREATED_FROM_CO_ID"/>
                </xs:keyref>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="MEDIA_OBJECTS" minOccurs="0">
          <xs:annotation>
            <xs:appinfo>
              <XDM>
                <Type value="struct"/>
              </XDM>
            </xs:appinfo>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="MO" type="xde:A_MEDIA_OBJECT" maxOccurs="unbounded">
                <xs:annotation>
                  <xs:appinfo>
                    <XDM>
                      <Map name="A_MEDIA_OBJECTS"/>
                      <Type value="exp"/>
                      <ImportAction action="useOld"/>
                      <ErrorAction action="addNew"/>
                      <Requirements>
                        <Required att="MO_PARENT_MO_ID" parentAtt="MO_ID"/>
                      </Requirements>
                      <Sequencers>
                        <Sequencer att="MO_ID" seq="A_OBJECTS_SEQ"/>
                      </Sequencers>
                    </XDM>
                  </xs:appinfo>
                </xs:annotation>
                <xs:key name="MO_ID">
                  <xs:selector xpath="."/>
                  <xs:field xpath="@MO_ID"/>
                </xs:key>
                <xs:keyref name="MO_MOT_ID" refer="xde:MOT_ID">
                  <xs:selector xpath="."/>
```

```
                    <xs:field xpath="@MO_MOT_ID"/>
                  </xs:keyref>
                  <xs:keyref name="MO_MT_ID" refer="xde:MT_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@MO_MT_ID"/>
                  </xs:keyref>
                  <xs:keyref name="MO_MMSV_ID" refer="xde:MMSV_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@MO_MMSV_ID"/>
                  </xs:keyref>
                  <xs:keyref name="MO_ASS_ID" refer="xde:ASS_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@MO_ASS_ID"/>
                  </xs:keyref>
                  <xs:keyref name="MO_MGSV_ID" refer="xde:MGSV_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@MO_MGSV_ID"/>
                  </xs:keyref>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="XVRML_TEMPLATES" minOccurs="0">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="struct"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:element name="TF" minOccurs="0" maxOccurs="unbounded">
                  <xs:annotation>
                    <xs:appinfo>
                      <XDM>
                        <Type value="exp"/>
                        <Map name="A_TEMPLATE_FOLDERS"/>
                        <ImportAction action="useOld"/>
                        <ErrorAction action="addNew"/>
                        <Requirements>
                          <Required att="TF_PARENT_FOLD_ID" parentAtt="TF_ID"/>
                        </Requirements>
                        <Sequencers>
                          <Sequencer att="TF_ID" seq="A_FOLDERS_SEQ"/>
                        </Sequencers>
                      </XDM>
                    </xs:appinfo>
                  </xs:annotation>
                  <xs:complexType>
                    <xs:complexContent>
                      <xs:extension base="xde:A_TEMPLATE_FOLDER"/>
                    </xs:complexContent>
                  </xs:complexType>
                  <xs:key name="TF_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@TF_ID"/>
                  </xs:key>
                </xs:element>
                <xs:element name="XVRML_TEMPLATE" type="xde:A_TEMPLATES" maxOccurs="unbounded">
                  <xs:annotation>
                    <xs:appinfo>
                      <XDM>
                        <Type value="exp"/>
                        <Map name="A_TEMPLATES"/>
                        <ImportAction action="useOld"/>
                        <ErrorAction action="addNew"/>
                        <Sequencers>
                          <Sequencer att="TPL_ID" seq="A_TEMPLATES_SEQ"/>
                        </Sequencers>
                      </XDM>
                    </xs:appinfo>
                  </xs:annotation>
                  <xs:key name="TPL_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@TPL_ID"/>
                  </xs:key>
                  <xs:keyref name="TPL_SUPERIOR_TEMPL_ID" refer="xde:TPL_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@TPL_SUPERIOR_TEMPL_ID"/>
                  </xs:keyref>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="TEMPL_OBJECTS" minOccurs="0">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="struct"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
```

```
            <xs:complexType>
              <xs:sequence>
                <xs:element       name="TOF"       type="xde:A_TEMPL_OBJ_FOLDER"       minOccurs="0"
maxOccurs="unbounded">
                  <xs:annotation>
                    <xs:appinfo>
                      <XDM>
                        <Type value="exp"/>
                        <Map name="A_TEMPL_OBJ_FOLDERS"/>
                        <ImportAction action="useOld"/>
                        <ErrorAction action="addNew"/>
                        <Requirements>
                          <Required att="TOF_TOF_ID" parentAtt="TOF_ID"/>
                        </Requirements>
                        <Sequencers>
                          <Sequencer att="TOF_ID" seq="A_FOLDERS_SEQ"/>
                        </Sequencers>
                      </XDM>
                    </xs:appinfo>
                  </xs:annotation>
                  <xs:key name="TOF_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@TOF_ID"/>
                  </xs:key>
                </xs:element>
                <xs:element name="TO" type="xde:A_TEMPL_OBJECT" maxOccurs="unbounded">
                  <xs:annotation>
                    <xs:appinfo>
                      <XDM>
                        <Type value="exp"/>
                        <Map name="A_TEMPL_OBJECTS"/>
                        <ImportAction action="useOld"/>
                        <ErrorAction action="addNew"/>
                        <Sequencers>
                          <Sequencer att="TO_ID" seq="A_OBJECTS_SEQ"/>
                        </Sequencers>
                      </XDM>
                    </xs:appinfo>
                  </xs:annotation>
                  <xs:key name="TO_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@TO_ID"/>
                  </xs:key>
                  <xs:unique name="TO_NAME">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@TO_NAME"/>
                  </xs:unique>
                  <xs:keyref name="TO_TOT_ID" refer="xde:TOT_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@TO_TOT_ID"/>
                  </xs:keyref>
                  <xs:keyref name="TO_MT_ID" refer="xde:TD_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@TO_MT_ID"/>
                  </xs:keyref>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="GENERAL_DATA" minOccurs="0">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="struct"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:element name="GD" maxOccurs="unbounded">
                  <xs:annotation>
                    <xs:appinfo>
                      <XDM>
                        <Map name="A_GENERAL_DATA"/>
                        <Type value="exp"/>
                        <ImportAction action="useOld"/>
                        <ErrorAction action="addNew"/>
                        <Sequencers>
                          <Sequencer att="GD_ID" seq="A_DATA_SEQ"/>
                        </Sequencers>
                      </XDM>
                    </xs:appinfo>
                  </xs:annotation>
                  <xs:complexType name="A_GENERAL_DATA">
                    <xs:attribute name="GD_ID" type="xde:AT_ID" use="required"/>
                    <xs:attribute name="GD_DT_CODE" type="xde:AT_CODE" use="required"/>
                    <xs:attribute name="GD_VALUE" type="xde:AT_SHORT_STRING" use="optional"/>
                  </xs:complexType>
                  <xs:key name="GD_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@GD_ID"/>
                  </xs:key>
                  <xs:keyref name="GD_DT_CODE" refer="xde:DT_CODE">
```

```
                    <xs:selector xpath="."/>
                    <xs:field xpath="@GD_DT_CODE"/>
                  </xs:keyref>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="STATIC_DATA" minOccurs="0">
      <xs:annotation>
        <xs:appinfo>
          <XDM>
            <Type value="struct"/>
          </XDM>
        </xs:appinfo>
      </xs:annotation>
      <xs:complexType>
        <xs:sequence>
          <xs:element name="CULTURAL_OBJECTS" minOccurs="0">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="struct"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:element name="CO_MDSV" maxOccurs="unbounded">
                  <xs:annotation>
                    <xs:appinfo>
                      <XDM>
                        <Type value="exp"/>
                        <Map name="A_CO_METADATA_SCHEMA_VERSIONS"/>
                        <ImportAction action="useOld"/>
                        <ErrorAction action="addNew"/>
                        <Sequencers>
                          <Sequencer att="CMSV_ID" seq="A_SCHEMAS_SEQ"/>
                        </Sequencers>
                      </XDM>
                    </xs:appinfo>
                  </xs:annotation>
                  <xs:complexType name="A_CO_METADATA_SCHEMA_VERSION">
                    <xs:sequence>
                      <xs:element name="SCHEMA_AO" type="xde:AT_XML_XSD">
                        <xs:annotation>
                          <xs:appinfo>
                            <XDM>
                              <Type value="attEl"/>
                              <Map name="CMSV_SCHEMA_AO"/>
                            </XDM>
                          </xs:appinfo>
                        </xs:annotation>
                      </xs:element>
                      <xs:element name="SCHEMA_CO" type="xde:AT_XML_XSD">
                        <xs:annotation>
                          <xs:appinfo>
                            <XDM>
                              <Type value="attEl"/>
                              <Map name="CMSV_SCHEMA_CO"/>
                            </XDM>
                          </xs:appinfo>
                        </xs:annotation>
                      </xs:element>
                      <xs:element name="SCHEMA_RO" type="xde:AT_XML_XSD">
                        <xs:annotation>
                          <xs:appinfo>
                            <XDM>
                              <Type value="attEl"/>
                              <Map name="CMSV_SCHEMA_RO"/>
                            </XDM>
                          </xs:appinfo>
                        </xs:annotation>
                      </xs:element>
                    </xs:sequence>
                    <xs:attribute name="CMSV_ID" type="xde:AT_ID" use="required"/>
                    <xs:attribute name="CMSV_ISSUED_ON" type="xde:AT_DATE" use="required"/>
                    <xs:attribute name="CMSV_VERSION" type="xde:AT_VERSION" use="optional"/>
                  </xs:complexType>
                  <xs:key name="CMSV_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@CMSV_ID"/>
                  </xs:key>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="MEDIA_OBJECTS" minOccurs="0">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
```

```
                    <Type value="struct"/>
                  </XDM>
                </xs:appinfo>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="MOT" type="xde:A_MEDIA_OBJECT_TYPE" maxOccurs="unbounded">
                    <xs:annotation>
                      <xs:appinfo>
                        <XDM>
                          <Type value="exp"/>
                          <Map name="A_MEDIA_OBJ_TYPES"/>
                          <ImportAction action="useOld"/>
                          <ErrorAction action="addNew"/>
                          <Sequencers>
                            <Sequencer att="MOT_ID" seq="A_TYPES_SEQ"/>
                          </Sequencers>
                        </XDM>
                      </xs:appinfo>
                    </xs:annotation>
                    <xs:key name="MOT_ID">
                      <xs:selector xpath="."/>
                      <xs:field xpath="@MOT_ID"/>
                    </xs:key>
                    <xs:keyref name="MOT_MDSS_ID" refer="xde:MMS_ID">
                      <xs:selector xpath="."/>
                      <xs:field xpath="@MOT_MDSS_ID"/>
                    </xs:keyref>
                  </xs:element>
                  <xs:element name="MO_MDS" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                      <xs:appinfo>
                        <XDM>
                          <Type value="exp"/>
                          <Map name="A_MO_METADATA_SCHEMAS"/>
                          <ImportAction action="useOld"/>
                          <ErrorAction action="addNew"/>
                          <Sequencers>
                            <Sequencer att="MMS_ID" seq="A_SCHEMAS_SEQ"/>
                          </Sequencers>
                        </XDM>
                      </xs:appinfo>
                    </xs:annotation>
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element          name="A_MO_METADATA_SCHEMA_VERSION"          minOccurs="0"
maxOccurs="unbounded">
                          <xs:annotation>
                            <xs:appinfo>
                              <XDM>
                                <Type value="exp"/>
                                <Map name="A_MO_METADATA_SCHEMA_VERSIONS"/>
                                <ImportAction action="useOld"/>
                                <ErrorAction action="addNew"/>
                                <Requirements>
                                  <Required att="MMSV_MDSS_ID" parentAtt="MMS_ID"/>
                                </Requirements>
                                <Sequencers>
                                  <Sequencer att="MMSV_ID" seq="A_SCHEMAS_SEQ"/>
                                </Sequencers>
                              </XDM>
                            </xs:appinfo>
                          </xs:annotation>
                          <xs:complexType>
                            <xs:sequence>
                              <xs:element name="MMSV_SCHEMA" type="xde:AT_XML_AMS">
                                <xs:annotation>
                                  <xs:appinfo>
                                    <XDM>
                                      <Type value="attEl"/>
                                    </XDM>
                                  </xs:appinfo>
                                </xs:annotation>
                              </xs:element>
                            </xs:sequence>
                            <xs:attribute name="MMSV_ID" type="xde:AT_ID" use="required"/>
                            <xs:attribute name="MMSV_VERSION" type="xde:AT_VERSION" use="required"/>
                            <xs:attribute name="MMSV_ISSUED_ON" type="xde:AT_DATE" use="required"/>
                          </xs:complexType>
                          <xs:key name="MMSV_ID">
                            <xs:selector xpath="."/>
                            <xs:field xpath="@MMSV_ID"/>
                          </xs:key>
                        </xs:element>
                      </xs:sequence>
                      <xs:attribute name="MMS_NAME" type="xde:AT_NAME" use="required"/>
                      <xs:attribute name="MMS_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
                      <xs:attribute name="MMS_ID" type="xde:AT_ID" use="required"/>
                      <xs:attribute name="MMS_CURRENT_MMSV_ID" type="xde:AT_ID" use="optional"/>
                    </xs:complexType>
                    <xs:key name="MMS_ID">
                      <xs:selector xpath="."/>
                      <xs:field xpath="@MMS_ID"/>
```

```
                    </xs:key>
                    <xs:keyref name="MMS_CURRENT_MMSV_ID" refer="xde:MMSV_ID">
                      <xs:selector xpath="."/>
                      <xs:field xpath="@MMS_CURRENT_MMSV_ID"/>
                    </xs:keyref>
                  </xs:element>
                  <xs:element name="MO_GEN_MDSV" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                      <xs:appinfo>
                        <XDM>
                          <Type value="exp"/>
                          <Map name="A_MO_GEN_MD_SCHEMA_VERSIONS"/>
                          <ImportAction action="useOld"/>
                          <ErrorAction action="addNew"/>
                          <Sequencers>
                            <Sequencer att="MGSV_ID" seq="A_SCHEMAS_SEQ"/>
                          </Sequencers>
                        </XDM>
                      </xs:appinfo>
                    </xs:annotation>
                    <xs:complexType>
                      <xs:sequence>
                        <xs:element name="MGSV_SCHEMA_GEN_MO" type="xde:AT_XML_AMS">
                          <xs:annotation>
                            <xs:appinfo>
                              <XDM>
                                <Type value="attEl"/>
                              </XDM>
                            </xs:appinfo>
                          </xs:annotation>
                        </xs:element>
                      </xs:sequence>
                      <xs:attribute name="MGSV_ID" type="xde:AT_ID" use="required"/>
                      <xs:attribute name="MGSV_VERSION" type="xde:AT_VERSION" use="required"/>
                      <xs:attribute name="MGSV_ISSUED_ON" type="xde:AT_DATE" use="required"/>
                    </xs:complexType>
                    <xs:key name="MGSV_ID">
                      <xs:selector xpath="."/>
                      <xs:field xpath="@MGSV_ID"/>
                    </xs:key>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="XVRML_TEMPLATES" minOccurs="0">
              <xs:annotation>
                <xs:appinfo>
                  <XDM>
                    <Type value="struct"/>
                  </XDM>
                </xs:appinfo>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="TD" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                      <xs:appinfo>
                        <XDM>
                          <Type value="exp"/>
                          <Map name="A_TEMPLATE_DOMAINS"/>
                          <ImportAction action="useOld"/>
                          <ErrorAction action="addNew"/>
                          <Sequencers>
                            <Sequencer att="TD_ID" seq="A_TEMPLATES_SEQ"/>
                          </Sequencers>
                        </XDM>
                      </xs:appinfo>
                    </xs:annotation>
                    <xs:complexType name="A_TEMPLATE_DOMAINS">
                      <xs:attribute name="TD_ID" type="xde:AT_ID" use="required"/>
                      <xs:attribute name="TD_NAME" type="xde:AT_NAME" use="required"/>
                      <xs:attribute name="TD_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
                    </xs:complexType>
                    <xs:key name="TD_ID">
                      <xs:selector xpath="."/>
                      <xs:field xpath="@TD_ID"/>
                    </xs:key>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="XSL_TEMPLATES" minOccurs="0">
              <xs:annotation>
                <xs:appinfo>
                  <XDM>
                    <Type value="struct"/>
                  </XDM>
                </xs:appinfo>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="XSL_TEMPLATE" type="xde:A_XSL_TEMPLATE" maxOccurs="unbounded">
                    <xs:annotation>
```

```
                    <xs:appinfo>
                      <XDM>
                        <Type value="exp"/>
                        <Map name="A_XSL_TEMPLATES"/>
                        <ImportAction action="useOld"/>
                        <ErrorAction action="addNew"/>
                        <Sequencers>
                          <Sequencer att="XT_ID" seq="A_TEMPLATES_SEQ"/>
                        </Sequencers>
                      </XDM>
                    </xs:appinfo>
                  </xs:annotation>
                  <xs:key name="XT_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@XT_ID"/>
                  </xs:key>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="TEMPL_OBJECTS" minOccurs="0">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="struct"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:element name="TOT" type="xde:A_TEMPL_OBJ_TYPE" maxOccurs="unbounded">
                  <xs:annotation>
                    <xs:appinfo>
                      <XDM>
                        <Type value="exp"/>
                        <Map name="A_TEMPL_OBJ_TYPES"/>
                        <ImportAction action="useOld"/>
                        <ErrorAction action="addNew"/>
                        <Sequencers>
                          <Sequencer att="TOT_ID" seq="A_TYPES_SEQ"/>
                        </Sequencers>
                      </XDM>
                    </xs:appinfo>
                  </xs:annotation>
                  <xs:key name="TOT_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@TOT_ID"/>
                  </xs:key>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="DATA_TYPES" minOccurs="0">
            <xs:annotation>
              <xs:appinfo>
                <XDM>
                  <Type value="struct"/>
                </XDM>
              </xs:appinfo>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:element name="DATA_TYPE" maxOccurs="unbounded">
                  <xs:annotation>
                    <xs:appinfo>
                      <XDM>
                        <Type value="exp"/>
                        <Map name="A_DATA_TYPES"/>
                        <ImportAction action="useOld"/>
                        <ErrorAction action="addNew"/>
                      </XDM>
                    </xs:appinfo>
                  </xs:annotation>
                  <xs:complexType name="A_DATA_TYPES">
                    <xs:attribute name="DT_CODE" type="xde:AT_CODE" use="required"/>
                    <xs:attribute name="DT_NAME" type="xde:AT_NAME" use="required"/>
                    <xs:attribute name="DT_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
                    <xs:attribute          name="DT_PATTERN"          type="xde:AT_DATA_TYPE_PATTERN"
use="optional"/>
                    <xs:attribute      name="DT_VALUE_CATEGORY"      type="xde:AT_DT_VALUE_CATEGORY"
use="required"/>
                  </xs:complexType>
                  <xs:key name="DT_CODE">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@DT_CODE"/>
                  </xs:key>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="MIME_TYPES" minOccurs="0">
            <xs:annotation>
              <xs:appinfo>
```

```
                  <XDM>
                    <Type value="struct"/>
                  </XDM>
                </xs:appinfo>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="MIMETYPE" maxOccurs="unbounded">
                    <xs:annotation>
                      <xs:appinfo>
                        <XDM>
                          <Type value="exp"/>
                          <Map name="A_MIMETYPES"/>
                          <ImportAction action="useOld"/>
                          <ErrorAction action="addNew"/>
                          <Sequencers>
                            <Sequencer att="MT_ID" seq="A_DATA_SEQ"/>
                          </Sequencers>
                        </XDM>
                      </xs:appinfo>
                    </xs:annotation>
                    <xs:complexType name="A_MIME_TYPE">
                      <xs:attribute name="MT_ID" type="xde:AT_ID" use="required"/>
                      <xs:attribute name="MT_NAME" type="xde:AT_NAME" use="required"/>
                      <xs:attribute      name="MT_EXTENSIONS"      type="xde:AT_MIME_TYPE_EXTENSIONS"
use="optional"/>
                      <xs:attribute name="MT_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
                    </xs:complexType>
                    <xs:key name="MT_ID">
                      <xs:selector xpath="."/>
                      <xs:field xpath="@MT_ID"/>
                    </xs:key>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="CONFIG_DATA" minOccurs="0">
              <xs:annotation>
                <xs:appinfo>
                  <XDM>
                    <Type value="struct"/>
                  </XDM>
                </xs:appinfo>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="CF_DATA" maxOccurs="unbounded">
                    <xs:annotation>
                      <xs:appinfo>
                        <XDM>
                          <Type value="exp"/>
                          <Map name="A_CONFIG_DATA"/>
                          <ImportAction action="useOld"/>
                          <ErrorAction action="addNew"/>
                          <Sequencers>
                            <Sequencer att="CO_ID" seq="A_DATA_SEQ"/>
                          </Sequencers>
                        </XDM>
                      </xs:appinfo>
                    </xs:annotation>
                    <xs:complexType name="A_CONFIG_DATA">
                      <xs:attribute name="CO_ID" type="xde:AT_ID" use="required"/>
                      <xs:attribute name="CO_NAME" type="xde:AT_NAME" use="required"/>
                      <xs:attribute name="CO_VALUE" type="xde:AT_SHORT_STRING" use="optional"/>
                    </xs:complexType>
                    <xs:key name="CO_ID_CONFIG">
                      <xs:selector xpath="."/>
                      <xs:field xpath="@CO_ID"/>
                    </xs:key>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
            <xs:element name="PROPERTIES" minOccurs="0">
              <xs:annotation>
                <xs:appinfo>
                  <XDM>
                    <Type value="struct"/>
                  </XDM>
                </xs:appinfo>
              </xs:annotation>
              <xs:complexType>
                <xs:sequence>
                  <xs:element name="PRO" maxOccurs="unbounded">
                    <xs:annotation>
                      <xs:appinfo>
                        <XDM>
                          <Type value="exp"/>
                          <Map name="A_PROPERTIES"/>
                          <ImportAction action="useOld"/>
                          <ErrorAction action="addNew"/>
                          <Sequencers>
                            <Sequencer att="PRO_ID" seq="A_OBJECTS_SEQ"/>
```

```
                        </Sequencers>
                      </XDM>
                    </xs:appinfo>
                  </xs:annotation>
                  <xs:complexType name="A_PROPERTY2">
                    <xs:attribute name="PRO_ID" type="xde:AT_ID" use="required"/>
                    <xs:attribute name="PRO_NAME" type="xde:AT_NAME" use="required"/>
                    <xs:attribute name="PRO_LEVEL" type="xde:AT_LEVEL" use="required"/>
                    <xs:attribute name="PRO_DT_CODE" type="xde:AT_CODE" use="optional"/>
                    <xs:attribute name="PRO_CATEGORY" type="xde:AT_PRO_CATEGORY" use="required"/>
                    <xs:attribute name="PRO_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
                  </xs:complexType>
                  <xs:key name="PRO_ID">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@PRO_ID"/>
                  </xs:key>
                  <xs:keyref name="PRO_DT_CODE" refer="xde:DT_CODE">
                    <xs:selector xpath="."/>
                    <xs:field xpath="@PRO_DT_CODE"/>
                  </xs:keyref>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="A_CREATION_DATE" type="xde:AT_DATE" use="required"/>
  <xs:attribute name="A_AUTHOR" type="xde:AT_SHORT_STRING" use="required"/>
  <xs:attribute name="A_SOURCE" type="xde:AT_DATA_SOURCE" use="required"/>
  <xs:attribute name="A_PRIMARY_CONTENTS" type="xde:AT_PRIMARY_CONTENTS" use="required"/>
 </xs:complexType>
</xs:element>
```