IST IST-2000-28336 **ARCO**

Augmented Representation of Cultural Objects

XML Metadata, Schemas and XSL

Report on the XML Interface Description between System Components

Author: PUE, UoS

Issued by: ARCO Consortium

Version: 1.0, Release

Date: 28-Oct-2002

Report Documentation Page

Report Documentation

Period	2
Distribution	Paper, Web Site, and Email
Work Package	WP6
Deliverable	D10
Security	Public
Project Number	IST-2000-28336
File Name	ARCO-D10-1.0-R-281002

Report Change Log

Version	Author(s)	Description	Date
0.1, Draft	O. Huminiecki (PUE)	Initial draft of ARCO XML Data Exchange (XDE) format	24-Sep-2002
0.2, Draft	Joseph Darcy (UoS)	Editing	10-Oct-2002
0.3, Draft	O. Huminiecki (PUE)	Editing	14-Oct-2002
0.4, Draft	K. Walczak (PUE)	Changed structure of the document, UML diagrams	16-Oct-2002
0.5, Draft	O. Huminiecki (PUE)	Editing; New sections 3.4, 3.5	17-Oct-2002
0.6, Draft	S. Strykowski (PUE)	Changes in Sections 1, 2, 3.1, 3.2	21-Oct-2002
0.7, Draft	K. Walczak (PUE)	Editing	22-Oct-2002
0.8, Draft	S. Strykowski (PUE)	Editing; Changes in Sections 3.3, 3.4, 3.5	23-Oct-2002
0.9, Draft	O. Huminiecki (PUE)	Editing	25-Oct-2002
0.10, Final Draft	K. Walczak (PUE)	Final Editing	25-Oct-2002
1.0, Release	N. Mourkoussis (UoS)	Release on the web site	28-Oct-2002

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

Glossary

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

Table of Contents

List of Figures and Tables

Summary

One of the key elements of the ARCO project is the use of XML technologies to enable data interoperability between ARCO system components and between the ARCO system and external systems and applications.

The purpose of this document is to describe the interfaces of the ARCO components and the XML Data Exchange Format (XDE) developed for the second prototype of the ARCO system.

1. Introduction

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 [1] 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 second 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 for data exchange and not for storage.

In this document, 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 described. Finally, the detailed description of the XDE Data Exchange Format is provided.

2. ARCO System Interfaces

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 1.

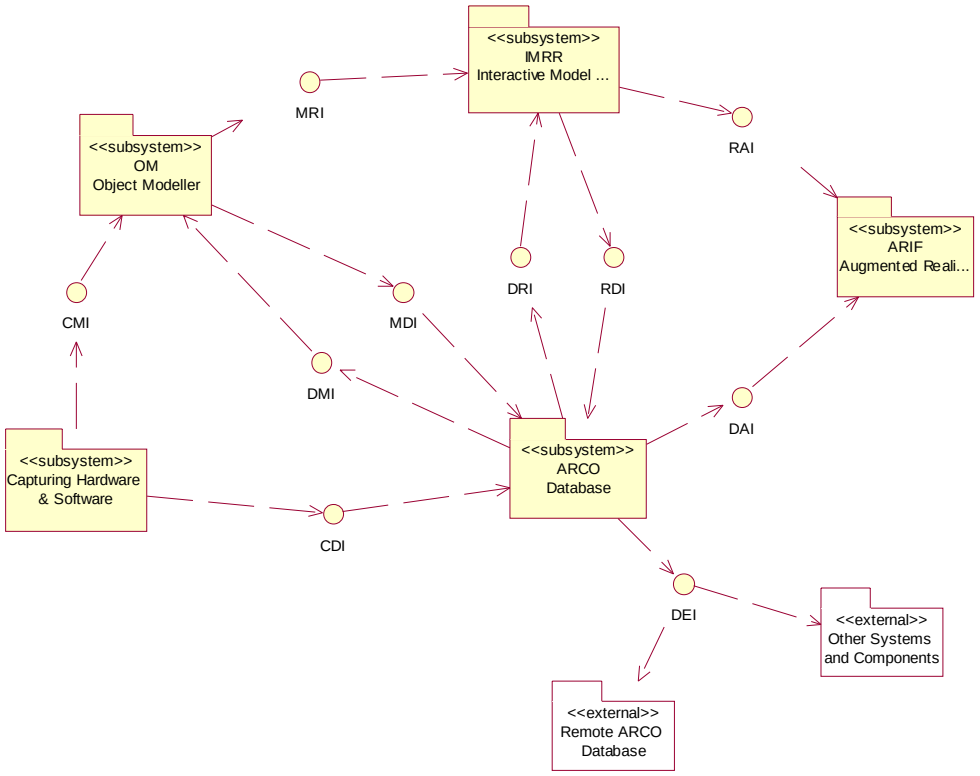

Figure 1. Interfaces of the ARCO system

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 second ARCO prototype: CDI, CMI, DMI, MDI, MRI, DRI, RDI, RAI, DAI, and DEI as shown in Figure 1. 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. More information on the interfaces can be found in [5].

This document contains description of all possible interfaces between ARCO system components. The fact that the 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. However, since this description serves the purpose of designing a common data exchange format for ARCO – XDE, all possibilities are taken into consideration.

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.

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 [2]. The images and metadata can be transferred to the ARCO database [6] using the *CDI* interface.

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.

2.3 Object Modeller

The Object Modeller is used to construct a 3D representation of an artefact from a set of images [9]. 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 material 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 3DS project file. Its sub-Media Objects of type Simple Image contain required materials images, as shown on Figure 2. The association between 3DS project MO and its sub-MOs, has an attribute which value is set to the relative path between 3DS materials directory and original location of images.

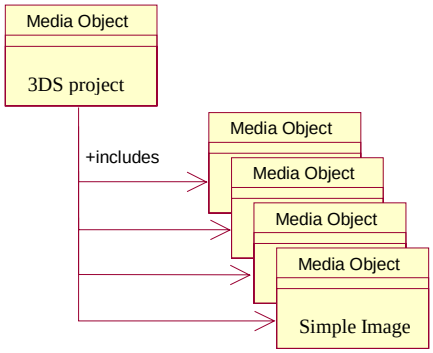

Figure 2. 3D Studio MAX Composite Media Object and its sub Media Objects

The composite Media Object of VRML model contains the VRML file. Its sub-Media Objects of Simple Image type contain required materials images, as shown on Figure 3. 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.

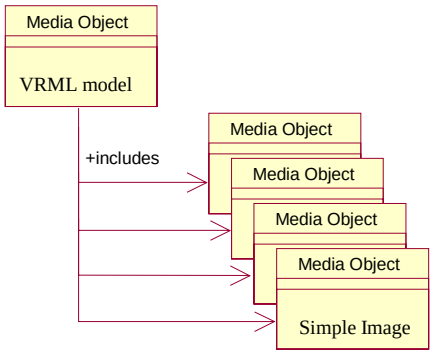

Figure 3. VRML Model Composite Media Object and its sub Media Objects

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.

2.4 Interactive Model Refinement and Rendering Tool

The Interactive Model Refinement and Rendering Tool – IMRR [7] is used to create the augmented representation of Cultural Objects originally created by the OM tool. It can be used to refine and/or to render the 3D representation of the artefact. 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.

2.5 Augmented Reality Interface – ARIF

The Augmented Reality Interface [8] is used to visualise the digital representations of cultural objects to the end users [4]. The content to be displayed in ARIF is dynamically created by the ARIF X-VRML Server (XTE) [3]. 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 ARIF tool by the use of the *RAI* interface.

2.6 ARCO Database

ARCO Database is a common repository for all kinds of data used in the ARCO system [6]. 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, as discussed in Section 1, exchange of 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.

3. XDE - ARCO XML Data Exchange Format

3.1 The Concept of XML Data Exchange Format

In Section 2, eleven different interfaces between ARCO system components are described. In order to avoid implementation of multiple different data formats for use over these interfaces, while 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 is produced and used by particular ARCO tools.

The exchange of data between system components is presented in Figure 4. 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.

Please note, that this document describes the rationale and design of the XDE file format and not the ARCO system architecture. The decision whether particular system interfaces will be implemented based on XDE or other communication medium is postponed for a later stage in the project.

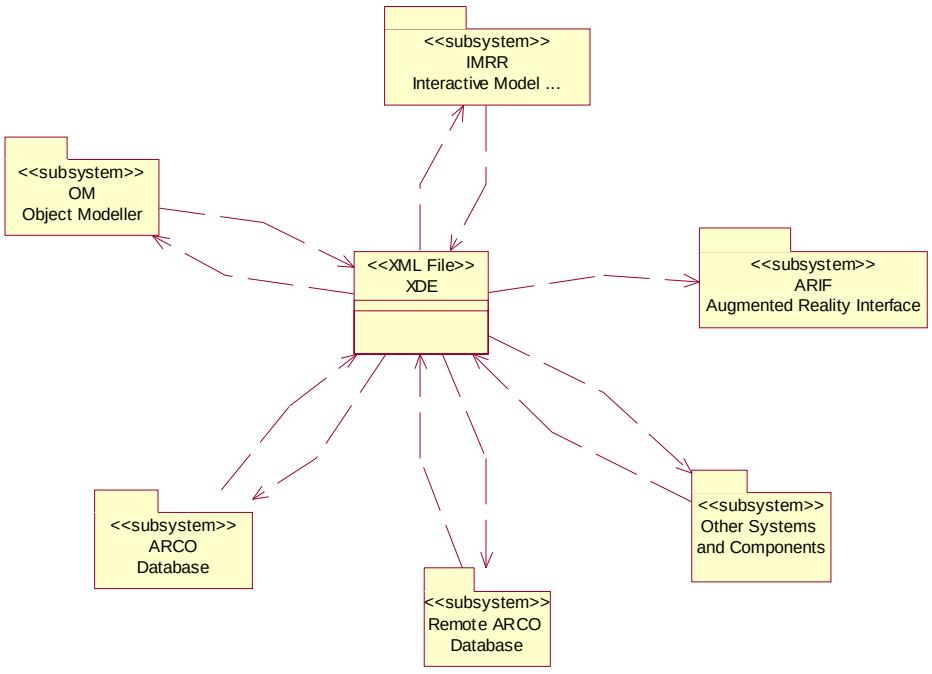

Figure 4. ARCO Data Exchange based on XDE

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.

3.2 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: `A_MEDIA_OBJECT`, `A_ASSOC_PARAM_VALUE`, `A_TEMPLATE_PARAM_VALUES`, and `A_TEMPL_OBJ_PARAM_VALUE` 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*.
    - Name of the child element used for modelling the many-to-many relationship is the same as *table3*;
    - 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:

    - `A_CULTURAL_OBJECT` to `A_MEDIA_OBJECT` with `A_CULTURAL_OBJ_TO_MEDIA_OBJ`;
    - `A_MEDIA_OBJECT_TYPE` to `A_MIME_TYPE` with `A_MEDIA_OBJ_TYPES_TO_MIME_TYPE`;
    - `A_CULTURAL_OBJ_FOLDER` to `A_CULTURAL_OBJECT` with `A_CULTURAL_OBJ_IN_FOLDER`;
    - `A_ARIF_FOLDER` to `A_CULTURAL_OBJ_FOLDER` with `A_CO_FOLDERS_IN_ARIF_FOLDER`
    - `A_ARIF_FOLDER` to `A_CULTURAL_OBJECT` with `A_CULTURAL_OBJ_IN_ARIF_FOLDER`;
    - `A_TEMPLATE_FOLDER` to `A_TEMPLATE` with `A_TEMPLATES_IN_FOLDER`;
    - `A_TEMPL_OBJ_FOLDER` to `A_TEMPLATE` with `A_TEMPL_OBJ_IN_FOLDER`;
    - `A_USER_GROUPS` to `A_PRIVILEGES` with `A_PRIVILEGES_FOR_USER_GROUPS`
    - `A_USER_GROUPS` to `A_USERS` with `A_USERS_IN_USERS_GROUPS`;

3.3 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 3.4 and 3.5.

3.3.1 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_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;
  - USERS – XDE file contains information about users and user 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_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

3.3.2 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 5. It contains two child elements corresponding to the two logical parts of the XDE content:

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

Figure 5. XDE root element - ARCO_DATA

The ARCO_DYNAMIC_DATA and ARCO_STATIC_DATA elements are described in Sections 3.4 and 3.5, respectively.

3.4 ARCO Dynamic Data

The structure of the ARCO_DYNAMIC_DATA element is shown in the Figure 6.

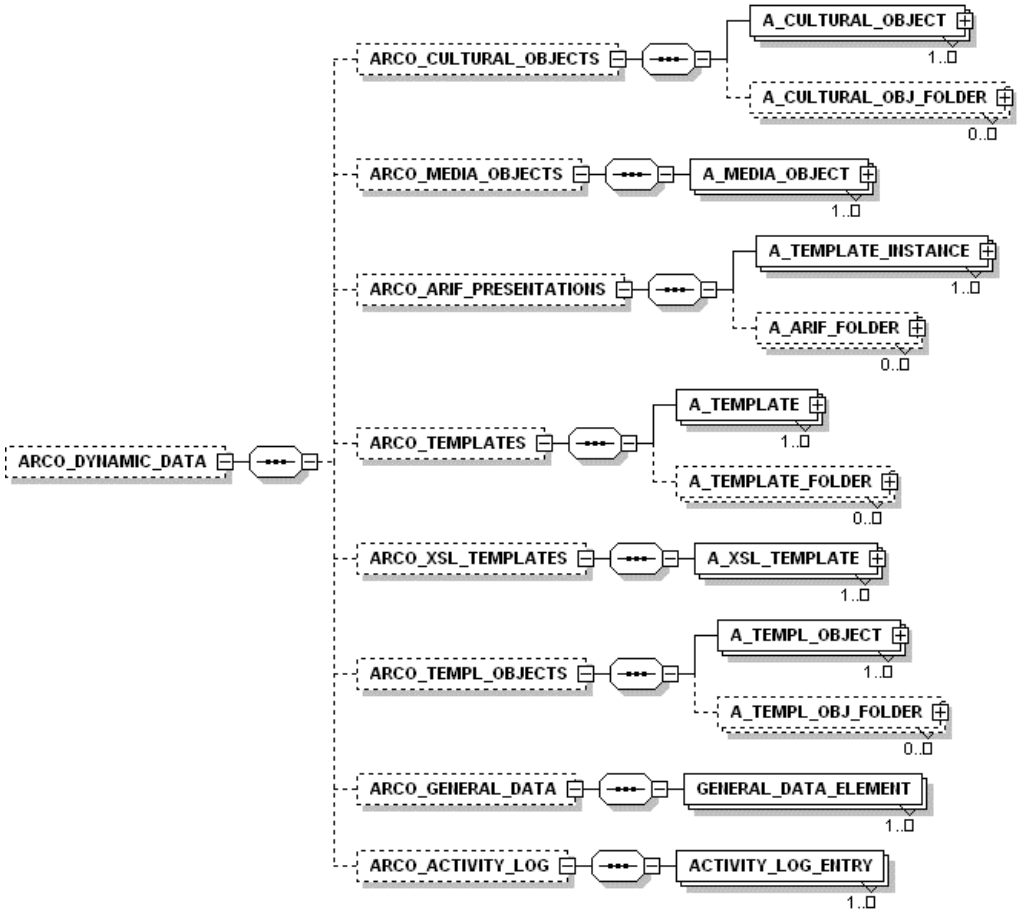

Figure 6. The structure of ARCO_DYNAMIC_DATA element

3.4.1 A_CULTURAL_OBJECT

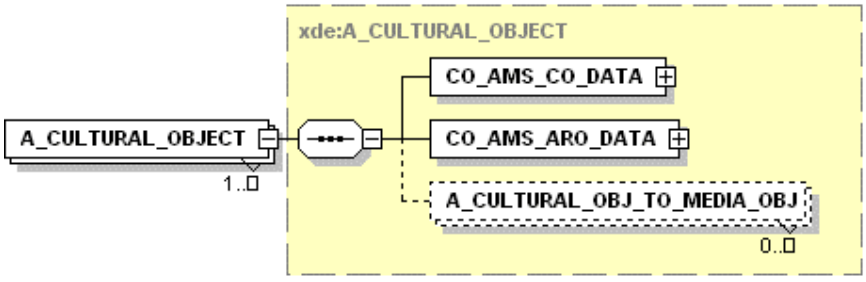

Figure 7. XML Schema of A_CULTURAL_OBJECT element

The `A_CULTURAL_OBJECT` element represents Cultural Object from the ARCO database (Figure 7). The optional and unbounded child element `A_CULTURAL_OBJ_TO_MEDIA_OBJ` has a key reference to the Media Object. This makes a vector of all Media Object associated with this Cultural Object. The `CO_AMS_CO_DATA` is the XML document containing Cultural Object AMS metadata. The `CO_AMS_ARO_DATA` is the XML document containing either Acquired Object or Refined Object AMS metadata depending of the `AT_CO_TYPE` attribute value.

Attributes:

Name	Type	Use
CO_ID	xde:AT_ID	required
CO_NAME	xde:AT_NAME	required
CO_CREATED_ON	xde:AT_DATE	required
CO_DESCRIPTION	xde:AT_DESCRIPTION	optional
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: A_CULTURAL_OBJ_TO_MEDIA_OBJ

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 A_CULTURAL_OBJECT element is shown in Figure 8, while its type definition in Figure 9.

```
<xs:element name="A_CULTURAL_OBJECT" type="xde:A_CULTURAL_OBJECT" maxOccurs="unbounded">
  <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 8. Implementation of A_CULTURAL_OBJECT element

```
<xs:complexType name="A_CULTURAL_OBJECT">
  <xs:sequence>
    <xs:element name="CO_AMS_CO_DATA" type="xde:AT_XML_AMS"/>
    <xs:element name="CO_AMS_ARO_DATA" type="xde:AT_XML_AMS"/>
    <xs:element name="A_CULTURAL_OBJ_TO_MEDIA_OBJ" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>This element contains attribute COMO_MO_ID pointing to
A_MEDIA_OBJECT</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:attribute name="COMO_MO_ID" type="xde:AT_DECIMAL" 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_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
  <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="AT_CO_TYPE" type="xde:AT_CO_TYPE" use="required"/>
</xs:complexType>
```

Figure 9. Implementation of A_CULTURAL_OBJECT type

3.4.2 A_CULTURAL_OBJ_FOLDER

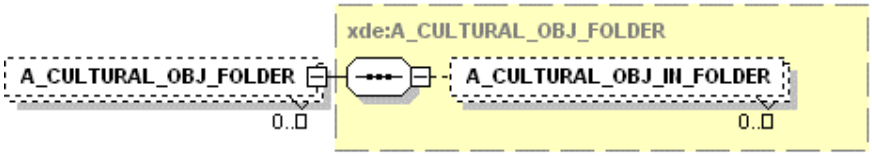

Figure 10. XML Schema of A_CULTURAL_OBJ_FOLDER element

The A_CULTURAL_OBJ_FOLDER element represents the hierarchical structure of Cultural Object Folders (Figure 10). The optional and unbounded element A_CULTURAL_OBJ_IN_FOLDER has a key reference to the Cultural Object 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_PARENT_FOLDER_ID	xde:AT_ID	optional

Constraints:

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

Child element: A_CULTURAL_OBJ_IN_FOLDER

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 A_CULTURAL_OBJ_FOLDER element is shown in Figure 11, while its type definition in Figure 12.

```
<xs:element name="A_CULTURAL_OBJ_FOLDER" type="xde:A_CULTURAL_OBJ_FOLDER" minOccurs="0"
maxOccurs="unbounded">
  <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 11. Implementation of A_CULTURAL_OBJ_FOLDER element

```
<xs:complexType name="A_CULTURAL_OBJ_FOLDER">
  <xs:sequence>
    <xs:element name="A_CULTURAL_OBJ_IN_FOLDER" minOccurs="0" maxOccurs="unbounded">
      <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: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:attribute name="COF_PARENT_FOLDER_ID" type="xde:AT_ID" use="optional"/>
</xs:complexType>
```

Figure 12. Implementation of A_CULTURAL_OBJ_FOLDER type

3.4.3 A_MEDIA_OBJECT

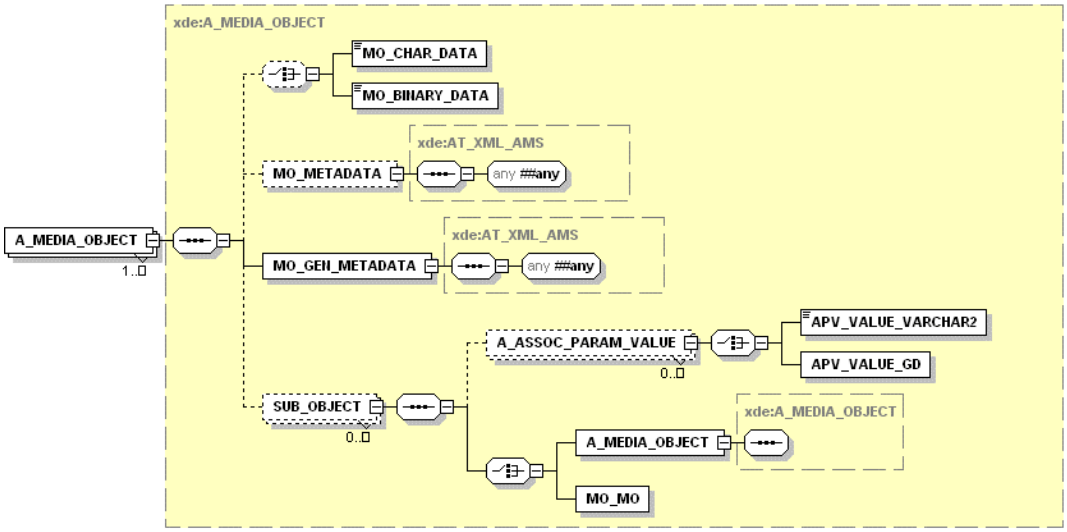

Figure 13. XML Schema of A_MEDIA_OBJECT element

The `A_MEDIA_OBJECT` element represents the Media Object from the ARCO database (Figure 13). Its structure reflects the structure of the composite Media Objects by joining Media Objects into a hierarchy with associations. The list of optional and unbounded `SUB_OBJECT` elements joins other Media Objects to this Media Object as its sub-objects. A `SUB_OBJECT` element may contain an unbounded list of `A_ASSOC_PARAM_VALUE` elements, which provide values of parameters of the association between the parent Media Object and the sub-Media Object. Each `A_ASSOC_PARAM_VALUE` has a key reference to the appropriate `A_ASSOCIATION_PARAMETER` definition, and a value. The `APV_VALUE_VARCHAR2` contains direct textual value, while the `APV_VALUE_GD` has key reference to `A_GENERAL_DATA`.

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
unique	MO_NAME		@MO_NAME
keyref	MO_ASS_ID	xde:ASS_ID	@MO_ASS_ID
keyref	MO_MGSV_ID	xde:MGSV_ID	@MO_MGSV_ID

Child element: A_MEDIA_OBJECT/SUB_OBJECT/MO_MO

Attributes:

Name	Type	Use
MO_MO_ID	xde:AT_ID	required

Constraints:

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

Child element: A_ASSOC_PARAM_VALUE

Attributes:

Name	Type	Use
APV_AP_ID	xde:AT_ID	required

Constraints:

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

Child element: A_ASSOC_PARAM_VALUE/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 A_MEDIA_OBJECT element is shown in Figure 14, while its type definition in Figure 15.

```
<xs:element name="A_MEDIA_OBJECT" type="xde:A_MEDIA_OBJECT" maxOccurs="unbounded">
  <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:unique name="MO_NAME">
    <xs:selector xpath="."/>
    <xs:field xpath="@MO_NAME"/>
  </xs:unique>
  <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 14. Implementation of A_MEDIA_OBJECT element

```
<xs:complexType name="A_MEDIA_OBJECT">
  <xs:sequence>
    <xs:choice minOccurs="0">
      <xs:element name="MO_CHAR_DATA" type="xde:AT_SHORT_STRING"/>
      <xs:element name="MO_BINARY_DATA" type="xde:AT_BINARY"/>
    </xs:choice>
    <xs:element name="MO_METADATA" type="xde:AT_XML_AMS" minOccurs="0"/>
    <xs:element name="MO_GEN_METADATA" type="xde:AT_XML_AMS"/>
    <xs:element name="SUB_OBJECT" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="A_ASSOC_PARAM_VALUE" minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
              <xs:documentation>This element contains attribute APV_AP_ID pointing to
A_ASSOC_PARAMETERS</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:choice>
                <xs:element name="APV_VALUE_VARCHAR2" type="xde:AT_SHORT_STRING"/>
                <xs:element name="APV_VALUE_GD">
                  <xs:complexType>
                    <xs:attribute name="APV_VALUE_GD_ID" type="xde:AT_ID"
use="required"/>
                  </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:complexType>
            <xs:keyref name="APV_AP_ID" refer="xde:AP_ID">
              <xs:selector xpath="."/>
              <xs:field xpath="@APV_AP_ID"/>
            </xs:keyref>
          </xs:element>
          <xs:choice>
            <xs:element name="A_MEDIA_OBJECT" type="xde:A_MEDIA_OBJECT"/>
            <xs:element name="MO_MO">
              <xs:annotation>
                <xs:documentation>This element contains attribute MO_MO_ID pointing to
A_MEDIA_OBJECT</xs:documentation>
              </xs:annotation>
              <xs:complexType>
                <xs:attribute name="MO_MO_ID" type="xde:AT_ID" use="required"/>
              </xs:complexType>
              <xs:keyref name="MO_MO_ID" refer="xde:MO_ID">
                <xs:selector xpath="."/>
                <xs:field xpath="@MO_MO_ID"/>
              </xs:keyref>
            </xs:element>
          </xs:choice>
        </xs:sequence>
      </xs:complexType>
    </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 15. Implementation of A_MEDIA_OBJECT type

3.4.4 A_TEMPLATE_INSTANCE

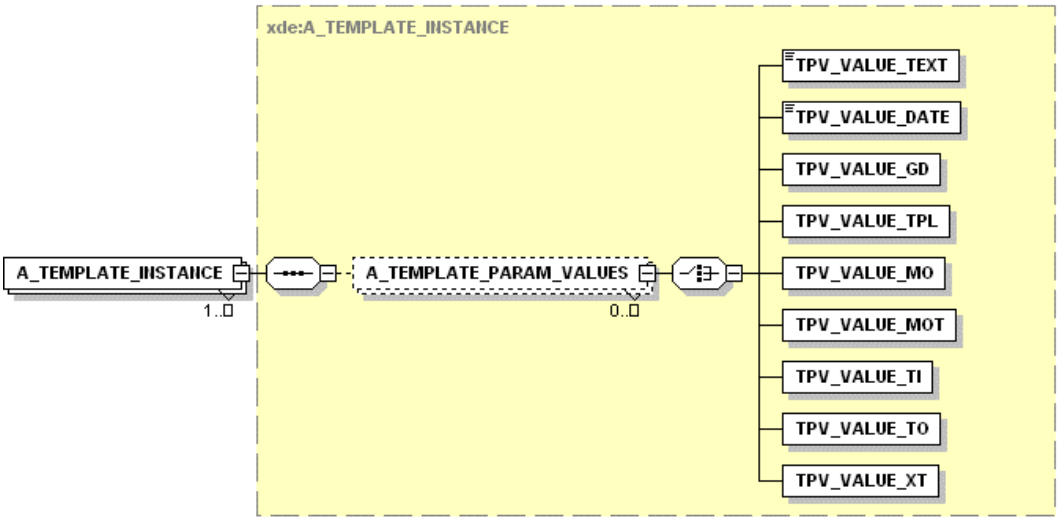

Figure 16. XML Schema of A_TEMPLATE_INSTANCE element

The `A_TEMPLATE_INSTANCE` element represents the X-VRML Template Instance (Figure 16). It has key reference `TI_TPL_ID` to the Template and another key reference `TI_AF_ID` to the ARIF Folder.

The optional and unbounded child element `A_TEPLATE_PARAM_VALUES` represents a set of Template Parameter values. Each `A_TEMPLATE_PARAM_VALUES` has a key reference `TPV_TP_ID` to the appropriate Template Parameter. The `A_TEMPLATE_PARAM_VALUES` has choice of child elements representing actual value of the parameter. `TPV_VALUE_TEXT` is a text value; `TPV_VALUE_DATE` is a date-and-time value. The `TPV_VALUE_GD`, `TPV_VALUE_TPL`, `TPV_VALUE_MO`, `TPV_VALUE_MOT`, `TPV_VALUE_TI`, `TPV_VALUE_TO`, and `TPV_VALUE_XT` are references to the General Data, Template, Media Object, Media Object Type, Template Instance, Template Object, and XSL Template, respectively.

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_AF_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_AF_ID	xde:AF_ID	@TI_AF_ID

Child element: A_TEMPLATE_PARAM_VALUES

Attributes:

Name	Type	Use
TPV_TP_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	TPV_TP_ID	xde:TP_ID	@TPV_TP_ID

The implementation of the A_TEMPLATE_INSTANCE element is shown in Figure 17, while its type definition in Figure 18.

```
<xs:element name="A_TEMPLATE_INSTANCE" type="xde:A_TEMPLATE_INSTANCE"
maxOccurs="unbounded">
  <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_AF_ID" refer="xde:AF_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TI_AF_ID"/>
  </xs:keyref>
</xs:element>
```

Figure 17. Implementation of A_TEMPLATE_INSTANCE element

```
<xs:complexType name="A_TEMPLATE_INSTANCE">
  <xs:sequence>
    <xs:element name="A_TEMPLATE_PARAM_VALUES" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>This element contains attribute TPV_TP_ID pointing to
A_TEMPLATE_PARAMS</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:choice>
          <xs:element name="TPV_VALUE_TEXT" type="xde:AT_STRING"/>
          <xs:element name="TPV_VALUE_DATE" type="xde:AT_DATE"/>
          <xs:element name="TPV_VALUE_GD">
            <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="TPV_VALUE_TPL">
            <xs:complexType>
              <xs:attribute name="TPV_VALUE_TPL_ID" type="xde:AT_ID" use="required"/>
            </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="TPV_VALUE_MO">
            <xs:complexType>
              <xs:attribute name="TPV_VALUE_MO_ID" type="xde:AT_ID" use="required"/>
            </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="TPV_VALUE_MOT">
            <xs:complexType>
              <xs:attribute name="TPV_VALUE_MOT_ID" type="xde:AT_ID" use="required"/>
            </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="TPV_VALUE_TI">
            <xs:complexType>
              <xs:attribute name="TPV_VALUE_TI_ID" type="xde:AT_ID" use="required"/>
            </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="TPV_VALUE_TO">
            <xs:complexType>
              <xs:attribute name="TPV_VALUE_TO_ID" type="xde:AT_ID" use="required"/>
            </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="TPV_VALUE_XT">
            <xs:complexType>
              <xs:attribute name="TPV_VALUE_XT_ID" type="xde:AT_ID" use="required"/>
            </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:attribute name="TI_AF_ID" type="xde:AT_ID" use="optional"/>
</xs:complexType>
```

Figure 18. Implementation of A_TEMPLATE_INSTANCE type

3.4.5 A_ARIF_FOLDER

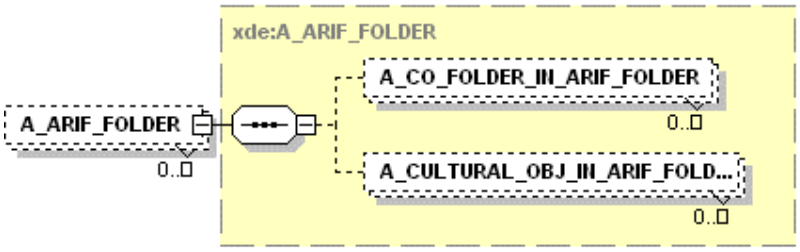

Figure 19. XML Schema of A_ARIF_FOLDER element

The A_ARIF_FOLDER element represents the ARIF Folder (Figure 19). The A_CO_FOLDER_IN_ARIF_FOLDER child element has a key reference to Cultural Object Folder. The A_CULTURAL_OBJ_IN_ARIF_FOLDER child element has key reference to the Cultural Object.

Attributes:

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

Constraints:

	Name	Refer	Field(s)
key	AF_ID		@AF_ID
keyref	AF_PARENT_FOLD_ID	xde:AF_ID	@AF_PARENT_FOLD_ID

Child element: A_CO_FOLDER_IN_ARIF_FOLDER

Attributes:

Name	Type	Use
CFAF_AF_ID	xde:AT_ID	required

Constraints:

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

Child element: A_CULTURAL_OBJ_IN_ARIF_FOLDER

Attributes:

Name	Type	Use
CAOF_AF_ID	xde:AT_ID	required

Constraints:

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

The implementation of the A_ARIF_FOLDER element is shown in Figure 20, while its type definition in Figure 21.

```
<xs:element name="A_ARIF_FOLDER" type="xde:A_ARIF_FOLDER" minOccurs="0"
maxOccurs="unbounded">
  <xs:key name="AF_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@AF_ID"/>
  </xs:key>
  <xs:keyref name="AF_PARENT_FOLD_ID" refer="xde:AF_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@AF_PARENT_FOLD_ID"/>
  </xs:keyref>
</xs:element>
```

Figure 20. Implementation of A_ARIF_FOLDER element

```
<xs:complexType name="A_ARIF_FOLDER">
  <xs:sequence>
    <xs:element name="A_CO_FOLDER_IN_ARIF_FOLDER" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="CFAF_AF_ID" type="xde:AT_ID"/>
      </xs:complexType>
      <xs:keyref name="CFAF_AF_ID" refer="xde:COF_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@CFAF_AF_ID"/>
      </xs:keyref>
    </xs:element>
    <xs:element name="A_CULTURAL_OBJ_IN_ARIF_FOLDER" minOccurs="0"
maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="CAOF_AF_ID" type="xde:AT_ID"/>
      </xs:complexType>
      <xs:keyref name="CAOF_AF_ID" refer="xde:CO_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@CAOF_AF_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:attribute name="AF_PARENT_FOLD_ID" type="xde:AT_ID" use="optional"/>
</xs:complexType>
```

Figure 21. Implementation of A_ARIF_FOLDER type

3.4.6 A_TEMPLATE

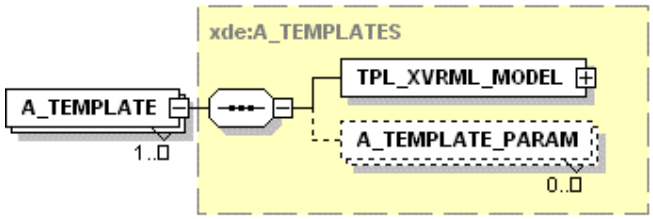

Figure 22. XML Schema of A_TEMPLATE element

The A_TEMPLATE element represents the X-VRML Template (Figure 22). It has the child element TPL_XVRML_MODEL containing the actual XML code of the X-VRML Template definition. The optional and unbounded child element A_TEMPLATE_PARAM is a list of Template parameters.

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

Constraints:

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

Child element: A_TEMPLATE_PARAM

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

The implementation of the A_TEMPLATE element is shown in Figure 23, while its type definition in Figure 24.

```
<xs:element name="A_TEMPLATE" type="xde:A_TEMPLATES" maxOccurs="unbounded">
  <xs:key name="TPL_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TPL_ID"/>
  </xs:key>
  <xs:keyref name="TPL_TD_ID" refer="xde:TD_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TPL_TD_ID"/>
  </xs:keyref>
  <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 23. Implementation of A_TEMPLATE element

```
<xs:complexType name="A_TEMPLATES">
  <xs:sequence>
    <xs:element name="TPL_XVRML_MODEL" type="xde:AT_XML_X-VRML"/>
    <xs:element name="A_TEMPLATE_PARAM" minOccurs="0" maxOccurs="unbounded">
      <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:sequence>
  <xs:attribute name="TPL_ID" type="xde:AT_ID" use="required"/>
  <xs:attribute name="TPL_TD_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:complexType>
```

Figure 24. Implementation of A_TEMPLATE type

3.4.7 A_TEMPLATE_FOLDER

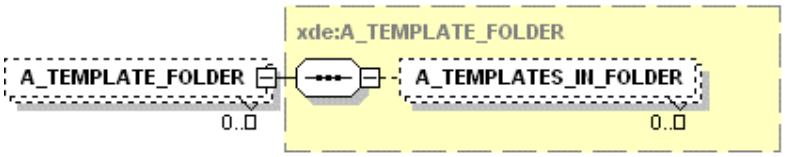

Figure 25. XML Schema of A_TEMPLATE_FOLDER element

The A_TEMPLATE_FOLDER element represents template folders (Figure 25). The unbounded and optional child element A_TEMPLATES_IN_FOLDER has key reference TINF_TPL_ID pointing to the X-VRML Template.

Attributes:

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

Constraints:

	Name	Refer	Field(s)
key	TF_ID		@TF_ID
keyref	TF_PARENT_FOLD_ID	xde:TF_ID	@TF_PARENT_FOLD_ID

Child element: A_TEMPLATE_PARAM_VALUES

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 A_TEMPLATE_FOLDER element is shown in Figure 26, while its type definition in Figure 27.

```
<xs:element name="A_TEMPLATE_FOLDER" type="xde:A_TEMPLATE_FOLDER" minOccurs="0"
maxOccurs="unbounded">
  <xs:key name="TF_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TF_ID"/>
  </xs:key>
  <xs:keyref name="TF_PARENT_FOLD_ID" refer="xde:TF_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TF_PARENT_FOLD_ID"/>
  </xs:keyref>
</xs:element>
```

Figure 26. Implementation of A_TEMPLATE_FOLDER element

```
<xs:complexType name="A_TEMPLATE_FOLDER">
  <xs:sequence>
    <xs:element name="A_TEMPLATES_IN_FOLDER" minOccurs="0" maxOccurs="unbounded">
      <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: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:attribute name="TF_PARENT_FOLD_ID" type="xde:AT_ID" use="optional"/>
</xs:complexType>
```

Figure 27. Implementation of A_TEMPLATE_FOLDER type

3.4.8 A_XSL_TEMPLATE

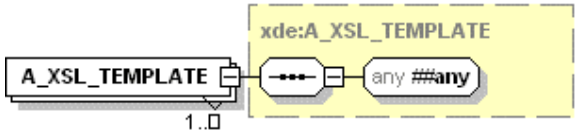

Figure 28. XML Schema of A_XSL_TEMPLATE element

The A_XSL_TEMPLATE element represents 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 A_XSL_TEMPLATE element is shown in Figure 29, while its type definition in Figure 30.

```
<xs:element name="A_XSL_TEMPLATE" type="xde:A_XSL_TEMPLATE" maxOccurs="unbounded">
  <xs:key name="XT_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@XT_ID"/>
  </xs:key>
</xs:element>
```

Figure 29. Implementation of A_XSL_TEMPLATE element

```
<xs:complexType name="A_XSL_TEMPLATE">
  <xs:complexContent>
    <xs:extension base="xde:AT_XML_XSL"/>
      <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:extension>
  </xs:complexContent>
</xs:complexType>
```

Figure 30. Implementation of A_XSL_TEMPLATE type

3.4.9 A_TEMPL_OBJECT

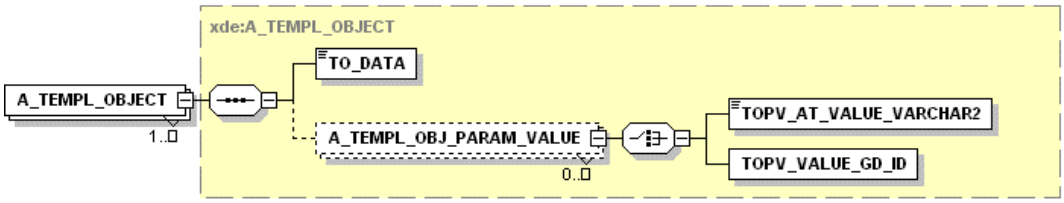

Figure 31. XML Schema of A_TEMPL_OBJECT element

The A_TEMPL_OBJECT element represents Template Object (Figure 31). The child element TO_DATA represents binary value of the Template Object, e.g. an image or a movie. The optional and unbounded element A_TEMPL_OBJ_PARAM_VALUE makes a list of template object parameters values. The A_TEMPL_OBJ_PARAM_VALUE 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: A_TEMPL_OBJ_PARAM_VALUE

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_ID

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 A_TEMPL_OBJECT element is shown in Figure 32, while its type definition in Figure 33.

```
<xs:element name="A_TEMPL_OBJECT" type="xde:A_TEMPL_OBJECT" maxOccurs="unbounded">
  <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 32. Implementation of A_TEMPL_OBJECT element

```
<xs:complexType name="A_TEMPL_OBJECT">
  <xs:sequence>
    <xs:element name="TO_DATA" type="xde:AT_BINARY"/>
    <xs:element name="A_TEMPL_OBJ_PARAM_VALUE" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>This element contains attribute TOPV_TOTP_ID pointing to
A_TEMPL_OBJ_TYPE_PARAMS</xs:documentation>
      </xs:annotation>
      <xs:complexType>
        <xs:choice>
          <xs:element name="TOPV_AT_VALUE_VARCHAR2" type="xde:AT_SHORT_STRING"/>
          <xs:element name="TOPV_VALUE_GD_ID">
            <xs:complexType>
              <xs:attribute name="ref" type="xde:AT_ID" use="required"/>
            </xs:complexType>
            <xs:keyref name="TOPV_VALUE_GD_ID" refer="xde:GD_ID">
              <xs:selector xpath="."/>
              <xs:field xpath="@ref"/>
            </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 33. Implementation of A_TEMPL_OBJECT type

3.4.10 A_TEMPL_OBJ_FOLDER

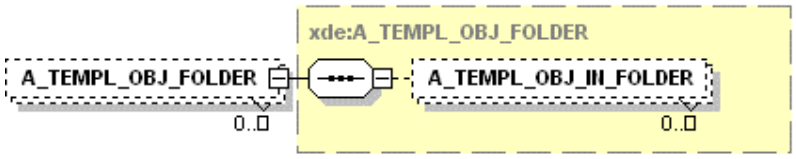

Figure 34. XML Schema of A_TEMPL_OBJ_FOLDER element

The A_TEMPL_OBJ_FOLDER element represents Template Object Folders (Figure 34). The unbounded and optional child element A_TEMPL_OBJ_IN_FOLDER 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: `A_TEMPL_OBJ_IN_FOLDER`

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 35, while its type definition in Figure 36.

```
<xs:element name="A_TEMPL_OBJ_FOLDER" type="xde:A_TEMPL_OBJ_FOLDER" minOccurs="0"
maxOccurs="unbounded">
  <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 35. Implementation of A_TEMPL_OBJ_FOLDER element

```
<xs:complexType name="A_TEMPL_OBJ_FOLDER">
  <xs:sequence>
    <xs:element name="A_TEMPL_OBJ_IN_FOLDER" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="TOF_TO_ID" type="xde:AT_ID" use="required"/>
      </xs:complexType>
      <xs:keyref name="TOF_TO_ID" refer="xde:TO_ID">
        <xs:selector xpath="."/>
        <xs:field xpath="@TOF_TO_ID"/>
      </xs:keyref>
    </xs:element>
  </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:attribute name="TOF_TOF_ID" type="xde:AT_ID" use="optional"/>
</xs:complexType>
```

Figure 36. Implementation of A_TEMPL_OBJ_FOLDER type

3.4.11 A_GENERAL_DATA_ELEMENT

Figure 37. XML Schema of GENERAL_DATA_ELEMENT element

The GENERAL_DATA_ELEMENT contains General Data values (Figure 37). Each GENERAL_DATA_ELEMENT element has its General Data Type indicated by 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 38.

```
<xs:element name="GENERAL_DATA_ELEMENT" maxOccurs="unbounded">
  <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 38. Implementation of GENERAL_DATA_ELEMENT element

3.4.12 A_ACTIVITY_LOG_ENTRY

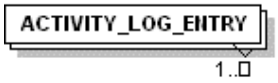

Figure 39. XML Schema of ACTIVITY_LOG_ENTRY element

The ACTIVITY_LOG_ENTRY represents an entry from user activity log (Figure 39). It contains the following information: activity event date, description and a key reference to the event type.

Attributes:

Name	Type	Use
AL_USER_ID	xde:AT_ID	required
AL_ACT_ID	xde:AT_ID	required
AL_TIMESTAMP	xde:AT_DATE	required
AL_REMARKS	xde:AT_DESCRIPTION	optional

Constraints:

	Name	Refer	Field(s)
keyref	AL_ACT_ID	xde:ACT_ID	@AL_ACT_ID
keyref	AL_USER_ID	xde:USR_ID	@AL_USER_ID

The implementation of the `ACTIVITY_LOG_ENTRY` element is shown in Figure 40.

```
<xs:element name="ACTIVITY_LOG_ENTRY" maxOccurs="unbounded">
  <xs:complexType name="A_ACTIVITY_LOG">
    <xs:attribute name="AL_USER_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="AL_ACT_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="AL_TIMESTAMP" type="xde:AT_DATE" use="required"/>
    <xs:attribute name="AL_REMARKS" type="xde:AT_DESCRIPTION" use="optional"/>
  </xs:complexType>
  <xs:keyref name="AL_ACT_ID" refer="xde:ACT_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@AL_ACT_ID"/>
  </xs:keyref>
  <xs:keyref name="AL_USER_ID" refer="xde:USR_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@AL_USER_ID"/>
  </xs:keyref>
</xs:element>
```

Figure 40. Implementation of ACTIVITY_LOG_ENTRY element

3.5 ARCO Static Data

The structure of the ARCO_STATIC_DATA element is shown in the Figure 41.

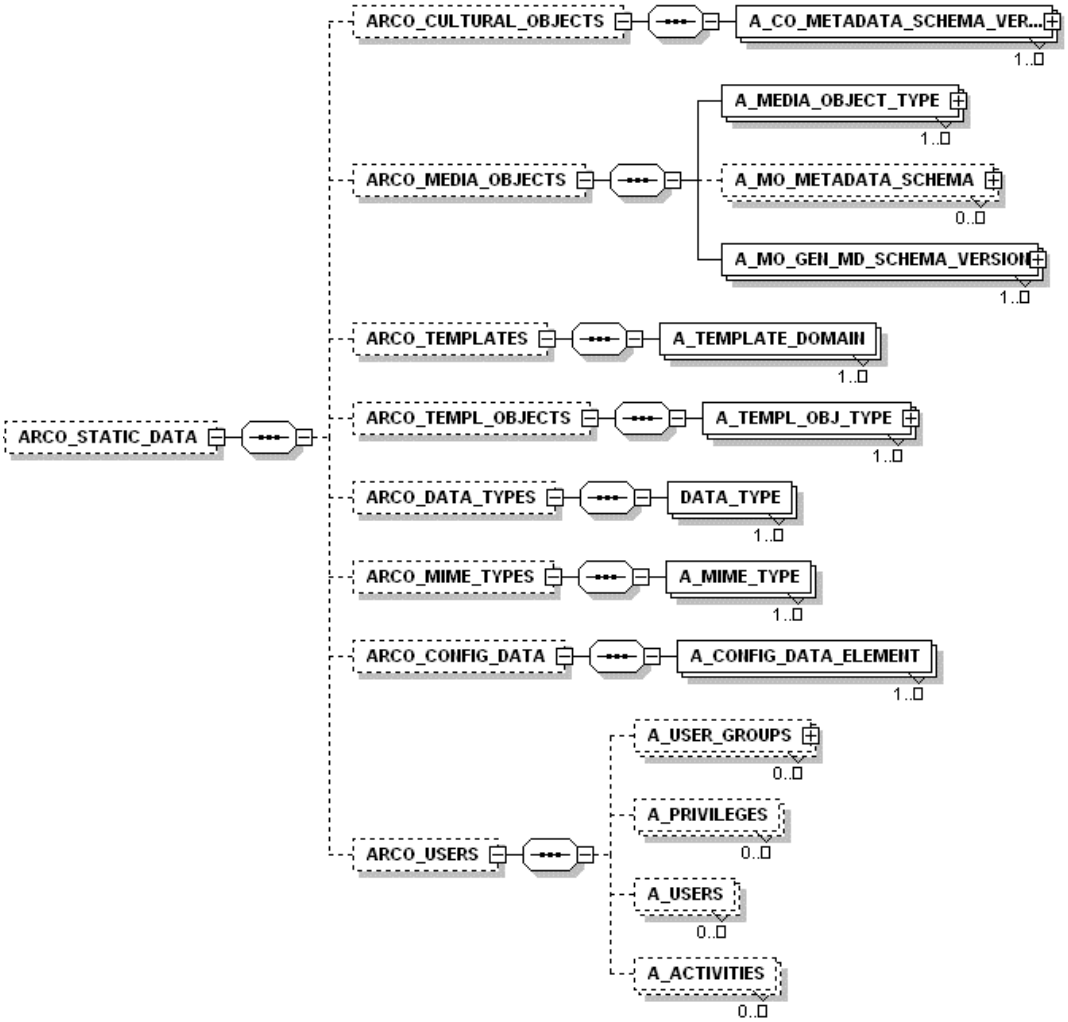

Figure 41. The structure of the ARCO_STATIC_DATA element

3.5.1 A_CO_METADATA_SCHEMA_VERSION

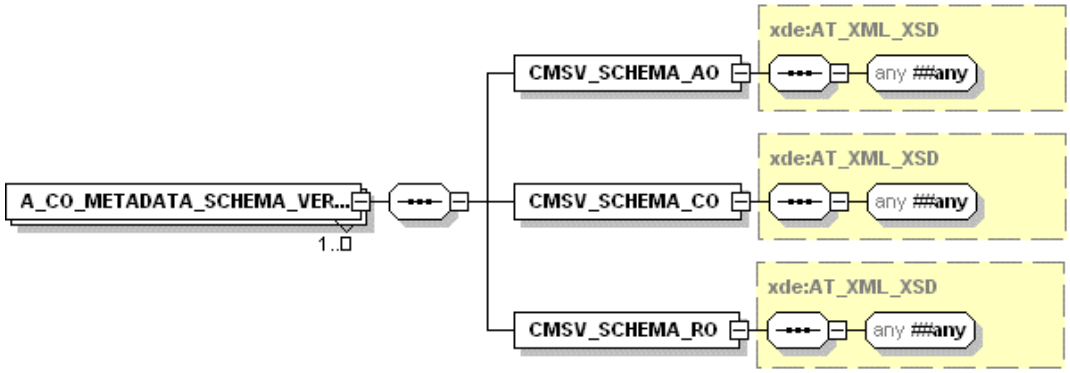

Figure 42. XML Schema of A_CO_METDATA_SCHEMA_VERSION element

The `A_CO_METDATA_SCHEMA_VERSION` element represents a version of Cultural Object AMS metadata schema (Figure 42). The `CMSV_SCHEMA_AO` child element carries Acquired Object AMS specification. The `CMSV_CHEMA_CO` child element carries general Cultural Object AMS specification. The `CMSV_CHEMA_RO` child element carries 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 `A_CO_METDATA_SCHEMA_VERSION` element is shown in Figure 43.

```
<xs:element name="A_CO_METADATA_SCHEMA_VERSION" maxOccurs="unbounded">
  <xs:complexType name="A_CO_METADATA_SCHEMA_VERSION">
    <xs:sequence>
      <xs:element name="CMSV_SCHEMA_AO" type="xde:AT_XML_XSD"/>
      <xs:element name="CMSV_SCHEMA_CO" type="xde:AT_XML_XSD"/>
      <xs:element name="CMSV_SCHEMA_RO" type="xde:AT_XML_XSD"/>
    </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 43. Implementation of A_CO_METDATA_SCHEMA_VERSION element

3.5.2 A_MEDIA_OBJECT_TYPE

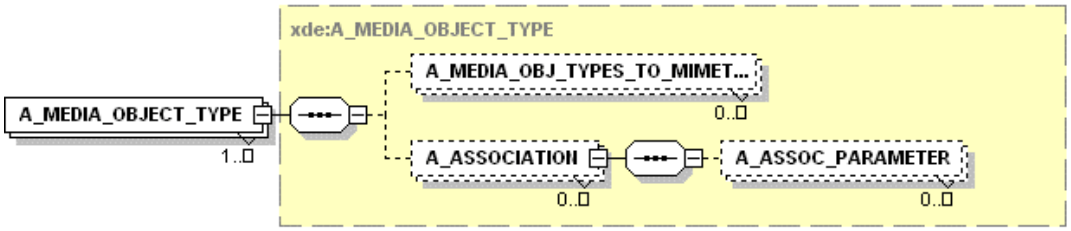

Figure 44. XML Schema of A_MEDIA_OBJECT_TYPE element

The `A_MEDIA_OBJECT_TYPE` element represents Media Object Type and the set of its associations (Figure 44). The optional and unbounded `A_MEDIA_OBJ_TYPES_TO_MIMETYPE` element points, by key reference `MTM_MT_ID`, to all related MIME-types. The `A_ASSOCIATION` element defines possible associations with other Media Object Types by key reference `ASS_MOT_ID`. The optional and unbounded child element `A_ASSOC_PARAMETER` 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: `A_MEDIA_OBJ_TYPES_TO_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: A_ASSOCIATION

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: A_ASSOCIATION/A_ASSOC_PARAMETER

Attributes:

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

Constraints:

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

The implementation of the A_MEDIA_OBJECT_TYPE element is shown in Figure 45, while its type definition in Figure 46.

```
<xs:element name="A_MEDIA_OBJECT_TYPE" type="xde:A_MEDIA_OBJECT_TYPE"
maxOccurs="unbounded">
  <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 45. Implementation of A_MEDIA_OBJECT_TYPE element

```
<xs:complexType name="A_MEDIA_OBJECT_TYPE">
  <xs:sequence>
    <xs:element name="A_MEDIA_OBJ_TYPES_TO_MIMETYPE" minOccurs="0"
maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="MTM_MT_ID" type="xs:decimal" 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="A_ASSOCIATION" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="A_ASSOC_PARAMETER" minOccurs="0" maxOccurs="unbounded">
            <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_ASS_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_ASS_ID" refer="xde:ASS_ID">
              <xs:selector xpath="."/>
              <xs:field xpath="@AP_ASS_ID"/>
            </xs:keyref>
            <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 46. Implementation of A_MEDIA_OBJECT_TYPE type

3.5.3 A_MO_METADATA_SCHEMA

Figure 47. XML Schema of A_MO_METADATA_SCHEMA element

The A_MO_METADATA_SCHEMA element represents Media Object AMS metadata schemas (Figure 47). The A_MO_METADATA_SCHEMA_VERSION element represents the specifications of Media Object schema versions. 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 A_MO_METADATA_SCHEMA element is shown in Figure 48.

```
<xs:element name="A_MO_METADATA_SCHEMA" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType name="A_MO_METADATA_SCHEMAS">
    <xs:sequence>
      <xs:element name="A_MO_METADATA_SCHEMA_VERSION" type="xde:AT_XML_XSD"
minOccurs="0" maxOccurs="unbounded">
        <xs:key name="MMSV_ID">
          <xs:selector xpath="."/>
          <xs:field xpath="@MMSV_ID"/>
        </xs:key>
        <xs:keyref name="MMSV_MDSS_ID" refer="xde:MMS_ID">
          <xs:selector xpath="."/>
          <xs:field xpath="@MMSV_MDSS_ID"/>
        </xs:keyref>
      </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 48. Implementation of A_MO_METADATA_SCHEMA element

3.5.4 A_MO_GEN_MD_SCHEMA_VERSION

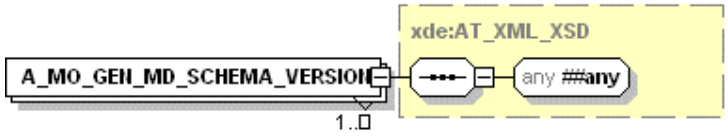

Figure 49. XML Schema of A_MO_GEN_MD_SCHEMA_VERSION element

The A_MO_GEN_MD_SCHEMA_VERSION element represents general Media Object AMS schema versions (Figure 49).

Constraints:

	Name	Refer	Field(s)
key	MGSV_ID		@MGSV_ID

The implementation of the A_MO_GEN_MD_SCHEMA_VERSION element is shown in Figure 50.

```
<xs:element name="A_MO_GEN_MD_SCHEMA_VERSION" type="xde:AT_XML_XSD"
maxOccurs="unbounded">
  <xs:key name="MGSV_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@MGSV_ID"/>
  </xs:key>
</xs:element>
```

Figure 50. Implementation of A_MO_GEN_MD_SCHEMA_VERSION element

3.5.5 A_TEMPLATE_DOMAIN

Figure 51. XML Schema of A_TEMPLATE_DOMAIN element

The A_TEMPLATE_DOMAIN element represents the set of possible X-VRML Template domains (Figure 51).

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)
unique	TD_ID		@TD_ID

The implementation of the A_TEMPLATE_DOMAIN element is shown in Figure 52.

```
<xs:element name="A_TEMPLATE_DOMAIN" maxOccurs="unbounded">
  <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:unique name="TD_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TD_ID"/>
  </xs:unique>
</xs:element>
```

Figure 52. Implementation of A_TEMPLATE_DOMAIN element

3.5.6 A_TEMPL_OBJ_TYPE

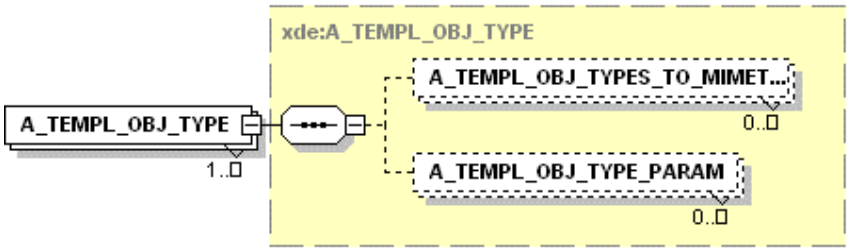

Figure 53. XML Schema of A_TEMPL_OBJ_TYPE element

The A_TEMPL_OBJ_TYPE element represents Template Object Type (Figure 53). The optional and unbounded child element A_TEMPL_OBJ_TYPES_TO_MIMETYPE element points, by key reference TOMT_MT_ID, to all related MIME-types. The optional and unbounded child element A_TEMPL_OBJ_TYPE_PARAMS element defines the set of Template Object 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: A_TEMPL_OBJ_TYPES_TO_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: A_TEMPL_OBJ_TYPE_PARAM

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 A_TEMPL_OBJ_TYPE element is shown in Figure 54, while its type definition in Figure 55.

```
<xs:element name="A_TEMPL_OBJ_TYPE" type="xde:A_TEMPL_OBJ_TYPE" maxOccurs="unbounded">
  <xs:key name="TOT_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@TOT_ID"/>
  </xs:key>
</xs:element>
```

Figure 54. Implementation of A_TEMPL_OBJ_TYPE element

```
<xs:complexType name="A_TEMPL_OBJ_TYPE">
  <xs:sequence>
    <xs:element name="A_TEMPL_OBJ_TYPES_TO_MIMETYPE" minOccurs="0"
maxOccurs="unbounded">
      <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="A_TEMPL_OBJ_TYPE_PARAM" minOccurs="0" maxOccurs="unbounded">
      <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 55. Implementation of A_TEMPL_OBJ_TYPE type

3.5.7 A_DATA_TYPE

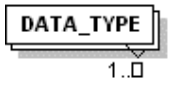

Figure 56. XML Schema of A_DATA_TYPE element

The DATA_TYPE element represents data type modelled in the ARCO database (Figure 56).

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 57.

```
<xs:element name="DATA_TYPE" maxOccurs="unbounded">
  <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 57. Implementation of A_DATA_TYPE element

3.5.8 A_MIME_TYPE

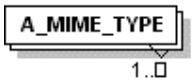

Figure 58. XML Schema of A_MIME_TYPE element

The A_MIME_TYPE element represents a MIME-type definition used across the ARCO content (Figure 58).

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 59.

```
<xs:element name="A_MIME_TYPE" maxOccurs="unbounded">
  <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_DESRIPTION" 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 59. Implementation of A_MIME_TYPE element

3.5.9 A_CONFIG_DATA_ELEMENT

Figure 60. XML Schema of A_CONFIG_DATA_ELEMENT element

The A_CONFIG_DATA_ELEMENT represents an ARCO system configuration parameter and its value (Figure 60).

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 A_CONFIG_DATA_ELEMENT element is shown in Figure 61.

```
<xs:element name="A_CONFIG_DATA_ELEMENT" maxOccurs="unbounded">
  <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>
```

Figure 61. Implementation of A_CONFIG_DATA_ELEMENT element

3.5.10 A_USER_GROUPS

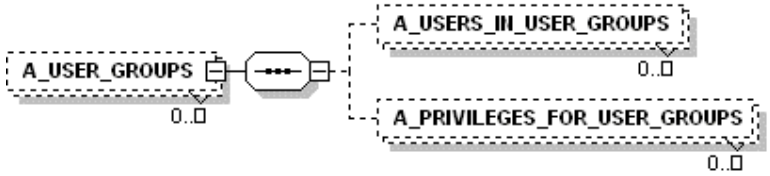

Figure 62. XML Schema of A_USER_GROUPS element

The A_USER_GROUPS element represents the ARCO system user groups (Figure 62). The optional and unbounded child element A_USERS_IN_USER_GROUPS point to group users by key reference UIUG_USER_ID. The optional and unbounded child element A_PRIVILEGES_FOR_USER_GROUPS points to group privileges by key reference PFUG_PRI_ID.

Attributes:

Name	Type	Use
UG_ID	xde:AT_ID	required
UG_NAME	xde:AT_NAME	required
UG_DESCRIPTION	xde:AT_DESCRIPTION	optional

Constraints:

	Name	Refer	Field(s)
key	UG_ID		@UG_ID

Child element: A_USERS_IN_USER_GROUPS

Attributes:

Name	Type	Use
UIUG_USER_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	UIUG_USER_ID	xde:USR_ID	@UIUG_USER_ID

Child element: A_PRIVILEGES_FOR_USER_GROUPS

Attributes:

Name	Type	Use
PFUG_PRI_ID	xde:AT_ID	required

Constraints:

	Name	Refer	Field(s)
keyref	PFUG_PRI_ID	xde:PRI_ID	@PFUG_PRI_ID

The implementation of the A_USER_GROUPS element is shown in Figure 63.

```
<xs:element name="A_USER_GROUPS" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType name="A_USER_GROUPS">
    <xs:sequence>
      <xs:element name="A_USERS_IN_USER_GROUPS" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="UIUG_USER_ID" type="xde:AT_ID" use="required"/>
        </xs:complexType>
        <xs:keyref name="UIUG_USER_ID" refer="xde:USR_ID">
          <xs:selector xpath="."/>
          <xs:field xpath="@UIUG_USER_ID"/>
        </xs:keyref>
      </xs:element>
      <xs:element name="A_PRIVILEGES_FOR_USER_GROUPS" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="PFUG_PRI_ID" type="xde:AT_ID" use="required"/>
        </xs:complexType>
        <xs:keyref name="PFUG_PRI_ID" refer="xde:PRI_ID">
          <xs:selector xpath="."/>
          <xs:field xpath="@PFUG_PRI_ID"/>
        </xs:keyref>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="UG_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="UG_NAME" type="xde:AT_NAME" use="required"/>
    <xs:attribute name="UG_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
  </xs:complexType>
  <xs:key name="UG_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@UG_ID"/>
  </xs:key>
</xs:element>
```

Figure 63. Implementation of A_USER_GROUPS element

3.5.11 A_PRIVILEGES

Figure 64. XML Schema of A_PRIVILEGES element

The `A_PRIVILEGES` element represents user's privileges (Figure 64).

Attributes:

Name	Type	Use
PRI_ID	xde:AT_ID	required
PRI_NAME	xde:AT_NAME	required
PRI_DESCRIPTION	xde:AT_DESCRIPTION	optional

Constraints:

	Name	Refer	Field(s)
key	PRI_ID		@PRI_ID

The implementation of the `A_PRIVILEGES` element is shown in Figure 65.

```
<xs:element name="A_PRIVILEGES" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType name="A_PRIVILEGES">
    <xs:attribute name="PRI_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="PRI_NAME" type="xde:AT_NAME" use="required"/>
    <xs:attribute name="PRI_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
  </xs:complexType>
  <xs:key name="PRI_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@PRI_ID"/>
  </xs:key>
</xs:element>
```

Figure 65. Implementation of A_PRIVILEGES element

3.5.12 A_USERS

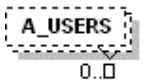

Figure 66. XML Schema of A_USRES element

The A_USERS element represents information about ARCO system users (Figure 66).

Attributes:

Name	Type	Use
USR_ID	xde:AT_ID	required
USR_NAME	xde:AT_NAME	required
USR_DESCRIPTION	xde:AT_DESCRIPTION	optional
USR_LOGIN_NAME	xde:AT_SHORT_STRING	required
USR_PASSWORD	xde:AT_PASSWORD	required
USR_CREATED_ON	xde:AT_DATE	required

Constraints:

	Name	Refer	Field(s)
key	USR_ID		@USR_ID

The implementation of the A_USRES element is shown in Figure 67.

```
<xs:element name="A_USERS" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType name="A_USERS">
    <xs:attribute name="USR_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="USR_NAME" type="xde:AT_NAME" use="required"/>
    <xs:attribute name="USR_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
    <xs:attribute name="USR_LOGIN_NAME" type="xde:AT_SHORT_STRING" use="required"/>
    <xs:attribute name="USR_PASSWORD" type="xde:AT_PASSWORD" use="required"/>
    <xs:attribute name="USR_CREATED_ON" type="xde:AT_DATE" use="required"/>
  </xs:complexType>
  <xs:key name="USR_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@USR_ID"/>
  </xs:key>
</xs:element>
```

Figure 67. Implementation of A_USRES element

3.5.13 A_ACTIVITES

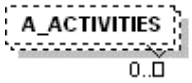

Figure 68. XML Schema of A_ACTIVITES element

The A_ACTIVITES element represents ARCO system user activity (Figure 68).

Attributes:

Name	Type	Use
ACT_ID	xde:AT_ID	required
ACT_NAME	xde:AT_NAME	required
ACT_DESCRIPTION	xde:AT_DESCRIPTION	optional

Constraints:

	Name	Refer	Field(s)
key	ACT_ID		@ACT_ID

The implementation of the A_ACTIVITES element is shown in Figure 69.

```
<xs:element name="A_ACTIVITIES" minOccurs="0" maxOccurs="unbounded">
  <xs:complexType name="A_ACTIVITIES">
    <xs:attribute name="ACT_ID" type="xde:AT_ID" use="required"/>
    <xs:attribute name="ACT_NAME" type="xde:AT_NAME" use="required"/>
    <xs:attribute name="ACT_DESCRIPTION" type="xde:AT_DESCRIPTION" use="optional"/>
  </xs:complexType>
  <xs:key name="ACT_ID">
    <xs:selector xpath="."/>
    <xs:field xpath="@ACT_ID"/>
  </xs:key>
</xs:element>
```

Figure 69. Implementation of A_ACTIVITES element

3.6 ARCO_DATA XML Schema Diagram

The XML Schema diagram of the XDE ARCO_DATA element is shown in Figure 70.

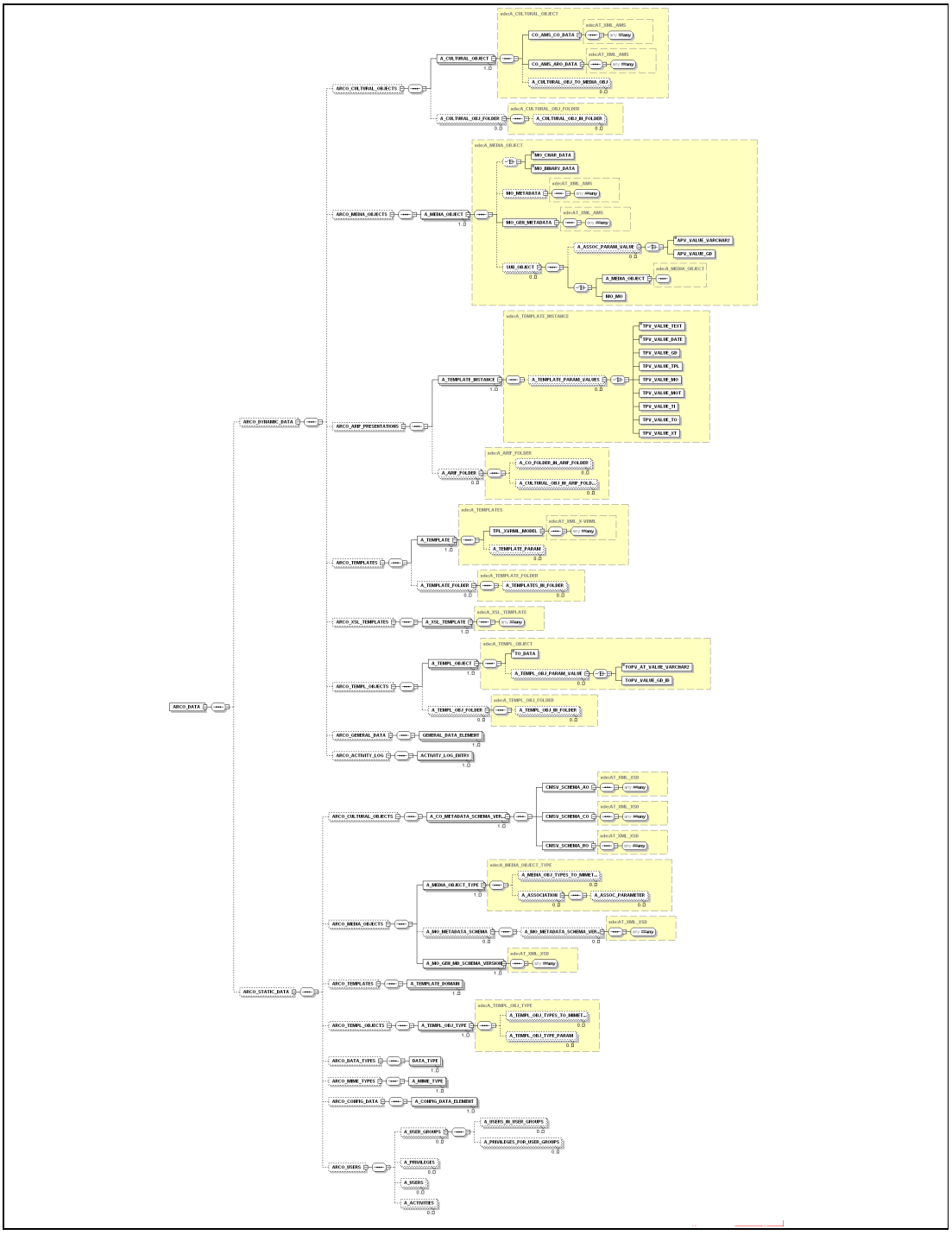

Figure 70. Structure of the XDE ARCO_DATA element

4. References

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

[2] ARCO Deliverable D8 – “Report on the XML descriptions of the database cultural objects”, ARCO Consortium 2002

[3] ARCO Deliverable D9 – “Report on XML Schemas, XSL Stylesheets and X-VRML Technology”, ARCO Consortium 2002

[4] ARCO Deliverable SSP-URS – "Second User Requirements Specification", Specification of the Second ARCO Prototype, ARCO Consortium 2002

[5] ARCO Deliverable SSP-SDS – “System Design Specification”, Specification of the Second ARCO Prototype, ARCO Consortium 2002

[6] ARCO Deliverable D3-SSP-ORDBMS – "Object Relational Database Management System", Specification of the Second ARCO Prototype, ARCO Consortium 2002

[7] ARCO Deliverable SSP-IMRR – “Interactive Model Refinement and Rendering tool”, Specification of the Second ARCO Prototype, ARCO Consortium 2002

[8] ARCO Deliverable D3-SSP-ARIF – "Augmented Reality Interface", Specification of the Second ARCO Prototype, ARCO Consortium 2002

[9] ARCO Deliverable SSP-OM – “Object Modeller”, Specification of the Second ARCO Prototype, ARCO Consortium 2002