ECP 2008 DILI 518002 EUscreen

Exploring Europe's Television Heritage in Changing Contexts

D4.2 - Report on the translation of EUscreen metadata on a semantic web language

Deliverable number	*D4.2 - Report on the translation of EUscreen metadata on a semantic web language*
Dissemination level	*Public*
Delivery date	*2010*
Status	*Final*
Author(s)	*Vassilis Tzouvaras (NTUA)*

e*Content*plus

This project is funded under the *e*Content*plus* programme[1]
a multiannual Community programme to make digital content in Europe more accessible, usable and exploitable.

[1] OJ L 79, 24.3.2005, p. 1.

Document Information

Deliverable number: *D4.2*
Deliverable title: *Report on the translation of EUscreen metadata on a semantic web language*
Actual date of deliverable:
Workpackage: 4
Workpackage title: Semantic Access & Integration
Workpackage leader: NTUA
Keywords: Semantic Web, Ontology Language, Metadata Standard

Table of Contents

1 Scope of this document

The aim of this deliverable is to report on the work done for the translation of the EUScreen schema (as this defined in the content selection strategy and metadata schema handbook of WP3) on a semantic web language. The output of this task is the EUScreen ontology that will be used for storing and searching the EUScreen metadata. The recent advances in Semantic Web technologies facilitate the way audiovisual archives (in general cultural heritage institutes) are representing their knowledge in a machine understandable language. In this way, web services can have access in the meaning of the information and provide useful services such as semantic search and alignment with external web resources using Lined-Open-Data technologies.

The deliverable is organized as follows. In chapter 2, the semantic web technologies available to be used in this task are presented. In chapter 3, the EUScreen ontology is presented. Finally, chapter 4 presents the concluding remarks and chapter 5 the references.

2 Semantic Web Technologies

2.1 ONTOLOGY DEFINITION LANGUAGES

An ontology is a controlled vocabulary that describes objects, and relations between them, in a formal way and has a grammar for using the vocabulary terms to express something meaningful, within a specified domain of interest. Ontologies in an application organize data used to describe other data, called metadata, in a machine understandable way giving the opportunity to agents to (semi)automatically carry out complex tasks assigned by humans in a meaningful (semantic) way. The most popular ontology definition languages are RDF, RDFS and OWL.

2.1.1 RDF

The Resource Description Framework [2][3] is a general-purpose language for representing information in the web. RDF's main elements are resources, properties and property values. A resource represents an object in our ontology which is connected through a property to some value which is either a literal or another resource. More than one resource can be interconnected and create a graph.

2.1.2 RDFS

RDFS (RDF Schema) [4] is an extension of RDF that is more expressible by allowing classes, as well as class and property subsumption. It provides mechanisms for describing groups of related resources and the relationships between these resources as well as other characteristic of resources, such as domain and range.

2.1.3 OWL

OWL is a Web Ontology Language [5]. OWL builds on RDF and RDFS and adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties and characteristics of properties (e.g. symmetry), and enumerated classes. It is also designed for use by applications that need to process the content of information instead of just presenting information to humans providing greater machine interpretability of Web content than that supported by RDF, and RDF Schema. OWL has three increasingly-expressive sublanguages: OWL Lite, OWL DL, and OWL Full.

OWL Lite

OWL Lite supports those users primarily needing a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It should be simpler to provide tool support for OWL Lite than its more expressive relatives, and OWL Lite provides a quick migration path for thesauri and other taxonomies. OWL Lite has a lower formal complexity than OWL DL.

OWL DL

OWL DL supports those users who want the maximum expressiveness while retaining computational completeness (all conclusions are guaranteed to be computable) and decidability (all computations will finish in finite time). OWL DL includes all OWL language constructs, but they can be used only under certain restrictions (for example, while a class may be a subclass of many classes, a class cannot be an instance of another class). OWL DL is so named due to its correspondence with description logics.

OWL Full

OWL Full is meant for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support complete reasoning for every feature of OWL Full.

OWL semantics are based on the formalism of Description Logics. OWL Lite and OWL DL are basically very expressive description logics almost equivalent to the SHIF(D+) and SHOIN(D+) Description Logics. Description Logics (DLs) [6] is the most recent name for a family of Knowledge Representation formalisms that represent the knowledge of an application domain by first defining the relevant concepts of the domain (its terminology), and then using this concepts to specify properties (called roles) of objects and individuals occurring in the domain (the world description). Typically we distinguish between atomic (or primitive) concepts, and complex concepts defined by using DL constructors. Different DL languages vary in the set of constructors provided. A DL Knowledge base comprises of two components, the TBox and the ABox. The TBox introduces the terminology, i.e. contains a set of concept descriptions and represents the general schema modeling the domain of interest. The ABox is a partial instantiation of this schema consisting of a set of assertions either relating individuals to classes, or individuals to each other.

One of the most attractive features of DLs is reasoning. Reasoning allows one to infer implicitly represented knowledge from the knowledge that is explicitly contained in the

knowledge base. Thus, we distinguish between TBox and ABox reasoning. Many of the applications only require reasoning in the TBox but in a demanding environment ABox reasoning is also essential. Reasoning tasks in a TBox are: satisfiability (consistency), that checks if a knowledge base is meaningful; subsumption, that checks whether all the individuals of a concept are subsumed (also belong) to another concept; equivalence, that checks whether two concepts denote the same set of instances; and disjointness, that checks whether the sets of instances of two concepts are disjoint. On the other hand reasoning tasks in ABox reasoning are: instance checking, that verifies whether an individual belongs to a given concept; consistency of the knowledge base, which checks whether the knowledge base is meaningful; and realization, that finds the most specific concept an individual object is an instance of.

2.2 Metadata standards

Metadata are data used to describe other data structured in formats easily understood by machines. One of the most familiar ways to organize metadata is through ontologies. Metadata standards are ontologies that define the vocabulary that describes the concepts and the relations among them in the specified domain of interest. The EUScreen project deals with cultural heritage content that is why in this section we will present some of the most important metadata.

2.2.1 MPEG-7

The MPEG-7 standard [7], formally named “Multimedia Content Description”, provides a rich set of standardized tools to describe multimedia content. MPEG-7 standardizes so-called “description tools” for multimedia content: Descriptors (Ds), Description Schemes (DSs) and the relationships between them. Descriptors are used to represent specific features of the content, generally low-level features such as visual (e.g. texture, camera motion) or audio (e.g. melody), while description schemes refer to more abstract description entities (usually a set of descriptors). These description tools as well as their relationships are represented using the Description Definition Language (DDL), a core part of the language. Both human users and automatic systems that process audiovisual information are within the scope of MPEG-7.

2.2.2 MPEG-21

The MPEG-21 standard [8] aims at defining a framework for multimedia delivery and consumption which supports a variety of businesses engaged in the trading of digital objects. The MPEG-21 standard is focusing on filling the gaps in the multimedia delivery chain. MPEG-21 was developed with the vision in mind that it should offer users transparent and interoperable consumption and delivery of rich multimedia content. The MPEG-21 standard

consists of a set of tools and builds on its previous coding and metadata standards like MPEG-1, -2, -4 and -7, i.e. it links them together to produce a protectable universal package for collecting, relating, referencing and structuring multimedia content for the consumption by users (the digital item). The vision of MPEG-21 is to enable transparent and augmented use of multimedia resources (e.g. Music tracks, videos, text documents or physical objects) contained in digital items across a wide range of networks and devices.

2.2.3 DUBLIN CORE

The Dublin Core Metadata Element Set [9] was proposed as a minimum number of metadata elements required to facilitate the creation of simple descriptive records for electronic documents. The set consists of a flat list of fifteen elements describing common properties of resources. To promote global interoperability, a number of the element descriptions may be associated with a controlled vocabulary for the respective element values. It is assumed that other controlled vocabularies will be developed for interoperability within certain local domains.

2.2.4 SPECTRUM

SPECTRUM documentation standard [10] is more than a metadata schema used mainly in museums. It is a guide to documenting all procedures a museum might need to undertake in managing its collections (e.g. acquisition, cataloguing, auditing, and loans). SPECTRUM recommends several "units of information" that can be recorded to support each of these procedures, some of which are required, others recommended. In terms of cataloguing museum objects, SPECTRUM suggests that sometimes it will be appropriate to catalogue at a collection level, at other times, at the item level. It suggests that any catalogue record should include at least: an identity number, name of the object, number of items or parts, physical description, and details about its acquisition, location and any associated images. SPECTRUM does not prescribe particular elements for digital reproductions, so those developing museum collection management systems often use SPECTRUM as the basis for the object information and Dublin Core to record information about any associated digital images.

2.2.5 CATEGORIES FOR THE DESCRIPTION OF WORKS OF ART (CDWA)

The CDWA [11] describes the content of art databases by articulating a conceptual framework for describing and accessing information about works of art, architecture, other material culture, groups and collections of works, and related images. The CDWA includes 381 categories and subcategories. A small subset of categories are considered core in that they represent the minimum information necessary to identify and describe a work.

2.2.6 ART MUSEUM IMAGE CONCORTIUM (AMICO)

The AMICO metadata vocabulary [12] is mainly used in the collection of art museum images. The AMICO metadata vocabulary using the DC and CDWA vocabularies provides a framework for the specification of images and multimedia files.

2.2.7 MACHINE-READABLE CATALOGUING (MARC21)

MARC21 [13] is a format standard for the storage and exchange of bibliographic records and related information in machine-readable form. Its format supported by the majority of library systems and offers participation in an international bibliographic community following common standards, and the advantage of copy cataloguing at much reduced cost and with no need to maintain conversion programs.

2.2.8 VISUAL RESOURCES ASSOCIATION (VRA) CORE

The Visual Resources Association (VRA) Core [14] is a standard that consists of a single metadata element set that can be applied to describe works of visual culture as well as the images that document them.

2.2.9 OPEN ARCHIVES INITIATIVE PROTOCOL FOR METADATA HARVESTING (OAI-PMH)

The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [15] provides an application-independent interoperability framework based on metadata harvesting. There are two classes of participants in the OAI-PMH framework: Data Providers administer systems that support the OAI-PMH as a means of exposing metadata; and Service Providers use metadata harvested via the OAI-PMH as a basis for building value-added services.

2.2.10 IMS

IMS [16] provides open technical specifications for interoperable learning technology and standards for delivering learning products and services. IMS collaborates with the organizations that develop and maintain profiles. Profiles are collections of one or more specifications/standards and extensions that an adopter community, such as a Governmental Department, or region has selected as a requirement for procurement.

2.2.11 SIMPLE KNOWLEDGE ORGANIZATION SYSTEM (SKOS)

Simple Knowledge Organization System (SKOS) Core [17] is a model and a RDF vocabulary for expressing the basic structure and content of concept schemes such as thesauri, classification schemes, subject heading lists, taxonomies, other types of controlled vocabulary, and also concept schemes embedded in glossaries and terminologies.

2.2.12 CIDOC-CONCEPTUAL REFERENCE MODEL (CRM)

CIDOC-CRM [18] is intended to promote a shared understanding of cultural heritage information by providing a common and extensible semantic framework to which any cultural heritage information can be mapped. It is intended to be a common language for domain experts and implementers to formulate requirements for information systems and to serve as a guide for good practice in conceptual modeling. In this way it can provide the "semantic glue" needed to mediate between different sources of cultural heritage information, such as that published by museums, libraries and archives.

2.2.13 INTERNATIONAL PRESS TELECOMMUNICATIONS COUNCIL (IPTC)

IPTC Core [19] is a set of metadata primarily for digital images to be used by Adobe's Extensible Metadata Platform XMP. IPTC metadata were employed by Adobe Systems Inc. to describe photos already in the early nineties. A subset of the IPTC "Information Interchange Model - IIM" was adopted as the well known "IPTC Headers" for Photoshop, JPEG and TIFF image files which currently describe millions of professional digital photos.

2.3 Ontology Stores

As described earlier ontologies play a key role in the representation of a knowledge base. However knowledge representation systems do not deal with large amount of information. They can only be applied to small knowledge bases, with a small number of instances, making reasoning algorithms not scalable and usually main memory oriented. On the other hand database systems provide technologies that deal with large, persistent and distributed amounts of information. The EUScreen project deals with a vast amount of metadata thus the combination of knowledge representation and database storing, called Semantic Web Storing Technologies, is essential for the storage of the semantic content. Semantic Web Storing Technologies combine inference services (reasoning), not only in the TBox but also in the ABox, with optimized storing models in order to make reasoning scalable and efficient.

In the following we present the state of the art on existing tools. In order to compare these tools we will focus on the following criteria, as described in [26]:

- Ontology Definition Language: The ontology definition language determines the expressiveness used to represent knowledge. Although this report focuses on RDF(S) and OWL storage, some tools store sublanguages of them, therefore expressiveness depends on the language operators that are supported.

- Storage Model: Most of the tools use relational technology to provide ontology persistence. We will study whether the ontology structure (TBox) is stored, and specifically, if a physical representation of the knowledge exists in the selected repositories, and if access paths (indexes) are implemented.

- Query Language: A Semantic Web query language should have features such as scalability and efficiency over distributed and massive storage data. Another important feature is the language expressiveness.

- Reasoning Mechanisms: It is particularly interesting to study the reasoning mechanisms that each tool supports, and how they are implemented. We study the TBox reasoning mechanism, but our main interest is the ABox reasoning mechanism assuming that the ABox is stored on disk (i.e. in a database), that is, not stored in main memory.

2.3.1 RDF(S) STORAGE MODELS

SESAME

SESAME [27] is an open source java framework for storing, querying and reasoning with RDF and RDFS.

Ontology Definition Language:

Ontologies are described using RDF.

Storage Model:

Sesame allows us to store RDF in several storage systems (Sesame repository) providing an API called SAIL. SAIL abstracts from the storage format used (i.e. whether the data is stored in an RDBMS, in memory or in files). Regarding databases, Sesame provides three implementations. In a first approach an ORDBMS has been used as a storage system, implemented using PostgreSQL. Later, a second approach using a relational database has been developed, and is implemented using MySQL. These storage models are broadly described in [27]. Currently it provides an implementation using Oracle 9i. The main objective of the proposed storage models is the separation of the RDF schema (information structure) and the RDF data (instances). Class instances are stored by means of unary relations and property instances are stored by means of binary relations.

Query Language:

The query language is RDQL [28], whose syntax is similar to SQL (it is therefore easy to use), and SeRQL [29] (that is currently being developed). Nevertheless, queries are translated to the query language of the corresponding storage device using the API. Therefore, it is not the storage device that performs the query, but the Sesame query engine.

Reasoning Mechanisms:

Sesame supports RDF schema inferencing by simply adding all implicit information to the repository as well when data is being added. Furthermore, tables to store property domains and ranges are defined to allow some inferences. For example, if we define a property instance P(x,y), we are able to infer that x is an instance of the class defined as domain of P, and y is an instance of the class defined as range of P. Inferencing depends on the repository being used. Not all the supported repositories allow inferencing.

JENA

JENA [30] is an open source project, implemented using java, which provides an API for manipulating RDF graphs. RDF ontologies are managed as graphs within the system. Jena provides some tools, such as a XML/RDF parser, a query language, and a storage module. Jena2 [31] is the second generation of Jena, and resolves Jena problems related to the ontologies storage.

Ontology Definition Language:

Ontologies are described using RDF. In Jena2 OWL ontologies are also handled.

Storage Model:

RDF graphs can be stored in the main memory or in a database. It is up to the user to decide where. RDF graph persistence is implemented by means of a relational database, and they are accessed using JDBC. The API supports several database management systems, like PostgresSQL, MySQL, Oracle, but it can be easily extended to support other management systems. Jena2 presents a denormalized storage schema.

Query Language:

The Jena query language is RDQL [28].

Reasoning Mechanisms:

RDQL does not support reasoning mechanisms. However, the sentences describing that a class is a sub-class of another class are stored in the database. Therefore, it is possible to compute the class hierarchy. On the other hand, Jena2 supports inference. The inference

subsystem is designed to allow a range of inference engines or reasoners to be plugged into Jena. Jena2 includes a number of predefined reasoners like the transitive reasoner, which implements the transitive and symmetric properties of the classes/properties hierarchy, the RDFS reasoner, which implements almost all the RDFS entailments, generic rule reasoner which supports rule-based inference over RDF graphs and provides forward chaining, backward chaining and a hybrid execution model and an OWL, OWL mini, OWL micro reasoners, which are incomplete implementations of OWL Lite.

KAON

KAON [32] is an open source infrastructure for managing ontologies. KAON provides an API that uncouples users from ontologies' persistent storage system. This API allows access to ontologies stored in a relational database.

Ontology Definition Language:

The KAON ontology query language is based on RDF. It contains some extensions, such as symmetric properties, transitive properties, inverse properties, modularization (reuse of concepts defined by other ontologies), meta-modeling (concepts that are instances of meta-concepts), and a lexical layer (information about ontology entities defined by the ontology itself). These extensions provide the language with part of the OWL expressiveness. Furthermore, defining properties as symmetric, transitive and inverse allows inferring instances of the properties non-explicitly defined.

Storage Model:

The Engineering Server is the API implementation that uses an ontology scalable representation in a relational database to store KAON ontologies. The server is optimized for KAON ontologies. These influence both the database's physical structure and API operations implementation. For example, the API supports transactional concept creation and deletion. Therefore, the database's physical schema contains a fixed number of tables (needed for transactional operations), and not one table for each concept or property, as in other approaches.

Query Language:

The KAON query language is in an experimental phase. The system provides a conceptual query language based on two main features: (1) Ontologies are interconnected object graphs; therefore navigation is the best way to explore them. (2) Ontologies are concepts and properties, which is why the queries' objective must be to define new concepts (intentional) and new properties (intentional). The language is based on the description logic systems

philosophy. In fact, the KAON query language is a description logic extended with some features needed to perform queries in a practical way, like the possibility of applying functions to concepts and properties. The language allows queries over both RDF schema and RDF data. Queries are translated to Datalog and they are evaluated in a deductive database.

Reasoning Mechanisms:

Reasoning mechanisms are implemented by the query language. They allow inferring instances from transitive, symmetric and inverse properties. This inference is handled by the API, processing the query and translating it to Datalog.

KOWARI

Kowari [33] is an open source, massively scalable, purpose-built database for the storage and retrieval of metadata. Therefore, it can be used to store and retrieve RDF. This repository can be used as a repository for the aforementioned systems. For example, it provides an API for RDQL, Jena, etc.

Ontology Definition Language:

Kowari supports RDF storage and retrieval.

Storage Model:

Kowari is not based on a relational database, it is a complete new database optimized for metadata management. Kowari is totally implemented in Java. It is optimized for storing short subject-predicate-object statements, providing native RDF support. The Statement Store stores statements as "quads" consisting of subject, predicate, object and metanodes. The first three items, from a standard RDF statement and the metanode describes which RDF model the statement appears in. More details about the Kowari storage model are given in [33]. Kowari also implements AVL indexes to improve queries that constraints one or more element of a triple (or quad if model information is added to a triple).

Query Language:

Kowari implements a simple SQLlike query language, the interactive Tucana Query language iTQL [34]. This language allows the creation of RDF models, the insertion or deletion of RFD statements in those models and the querying of results. iTQL supports queries based in constraint that limits individual subject-predicate-object triples, to specified patterns.

Reasoning Mechanisms:

Although is possible to implement almost all OWL constructors using iTQL, it is still ongoing work. Currently, it is possible to compute the class/property hierarchy.

2.3.2 OWL STORAGE MODELS

DLDB

DLDB [35] is a knowledge base that extends a relational database management system (RDBMS) with additional capabilities for making inferences. The main objective of this system is to study how description logic reasoning mechanisms can be combined with an RDBMS, in order to support extensional queries over OWL-DL documents.

Ontology Definition Language:

The language used to define ontologies is OWL-DL. Nevertheless, the proposed storage method is for RDF.

Storage Model:

Ontologies are stored using Microsoft Access as RDBMS. Specifically, the system stores RDF in a relational database. Ontologies are stored creating a table for each class or property definition. The class hierarchy is stored in the system using views. The view of a class is defined recursively, and consists of the union of its table and all the views of its direct sub-classes. Therefore, the view of a class includes the instances of that class plus the inferred instances using the taxonomic reasoning mechanism. The sub-property relationship is stored in a similar way. The system is optimized for medium size ontologies (hundreds of classes and properties).

Query Language:

The system provides an API, implemented in java, to query the database. It supports conjunctive queries in a format similar to KIF [36]. The query is translated to SQL and is sent to the database using JDBC. The query is evaluated by the RDBMS that returns the results.

Reasoning Mechanisms:

The reasoning mechanisms are implemented using the FaCT reasoner [37] coupled to the RDBMS. FaCT only supports TBox reasoning; therefore DLDB only implements those reasoning mechanisms that can be reduced to concept subsumption (concept and property taxonomy reasoning). These reasoning mechanisms are implemented by pre-computing the class/property hierarchy and storing it in the database using views. Using FaCT, we obtain the sub-classes/sub-properties of a given class/property needed to generate the views. The system

does not support other OWL-DL reasoning mechanisms, such as inverseOf, equivalentClass, hasValue, etc.

DLP-IM

The DLP implementation system (DLP-IM) [38] is a proposal for combining rules with ontologies in the Semantic Web. It is based on logical databases, since they provide a declarative knowledge representation language and persistent data storage. DLP define a mapping between a DL subset and Logic Programs (LP). This intersection between DL and LP, called DLP (Description Logic Programs), covers RDF schema completely, and part of OWL. In [39] an alternative mapping with less computational complexity is presented. This approach allows for greater representation flexibility. Following the nomenclature defined in, we refer to the first correspondence as direct correspondence and to the second one as metacorrespondence. In [39] is also presented a system implementing DLP.

Ontology Definition Language:

Ontologies are described by a subset of OWL, the subset that has a correspondence with LP.

Storage Model:

Ontologies are stored by translating them to LP. In direct correspondence, each class or property definition corresponds to a rule, and each class or property instance to a fact. In meta correspondence a meta-level is defined. Class and property names are meta-predicates. This meta-level consists of a set of facts representing the ontology content. The storage model is determined by Datalog. The system uses the deductive database CORAL [40].

Query Language:

The query language is Datalog.

Reasoning Mechanisms:

The reasoning mechanisms are those translatable to Datalog rules. Some ABox reasoning mechanisms that are relevant for Semantic Web applications are implemented by translating them into Datalog rules. For example, intersection, union, range (if an instance I is in the range of a relation with a range restriction on a class C, then I is an instance of C) and domain (if an instance I is in the domain of a relation with a domain restriction on class C, then I is an instance of C).

INSTANCE STORE (IS)

Instance Store (IS) [41] is an approach to a restricted form of ABox reasoning that combines a description logic reasoner with a database. The Instance Store can only deal with free-role ABox, i.e. ABox that do not contain any axioms asserting role relationships between pairs of individuals.

Ontology Definition Language:

Ontologies are described using OWL.

Storage Model:

Instance Store only offers persistence to the ABox. ABox assertions are stored in a relational database. An identifier, ids, is assigned to each description (concept) and a table stores individuals and the ids of their associated description. Another table contains description ids and all the primitive concepts in the ontology which subsume them. The primitive concepts which are equivalent to, parent of and child of a given description are also stored in a table.

Query Language:

Instance Store provides an API written in Java. This API contains a retrieval method that retrieves all the instances of a given concept. The query is translated to SQL and is sent to the relational database.

Reasoning Mechanisms:

Instance Store does not provide TBox reasoning mechanisms. That is, it is not possible to reason about the structure of the ontology. The only ABox mechanism is, as we said above, instance retrieval.

3 EUScreen schema in Semantic Web Language

3.1 Graphical Representation

The EUScreen ontology has been created using the Protégé Ontology Editing Tool [16]. The conceptualisation of the ontology has been made using the EUScreen content selection and metadata Handbook created in the framework of WP3. In the handbook all the fields required to annotate an EUScreen item are presented as well as the corresponding mapping to EBUcore schema is provided. In the figure below (3.1), the class hierarchy of the ontology is presented.

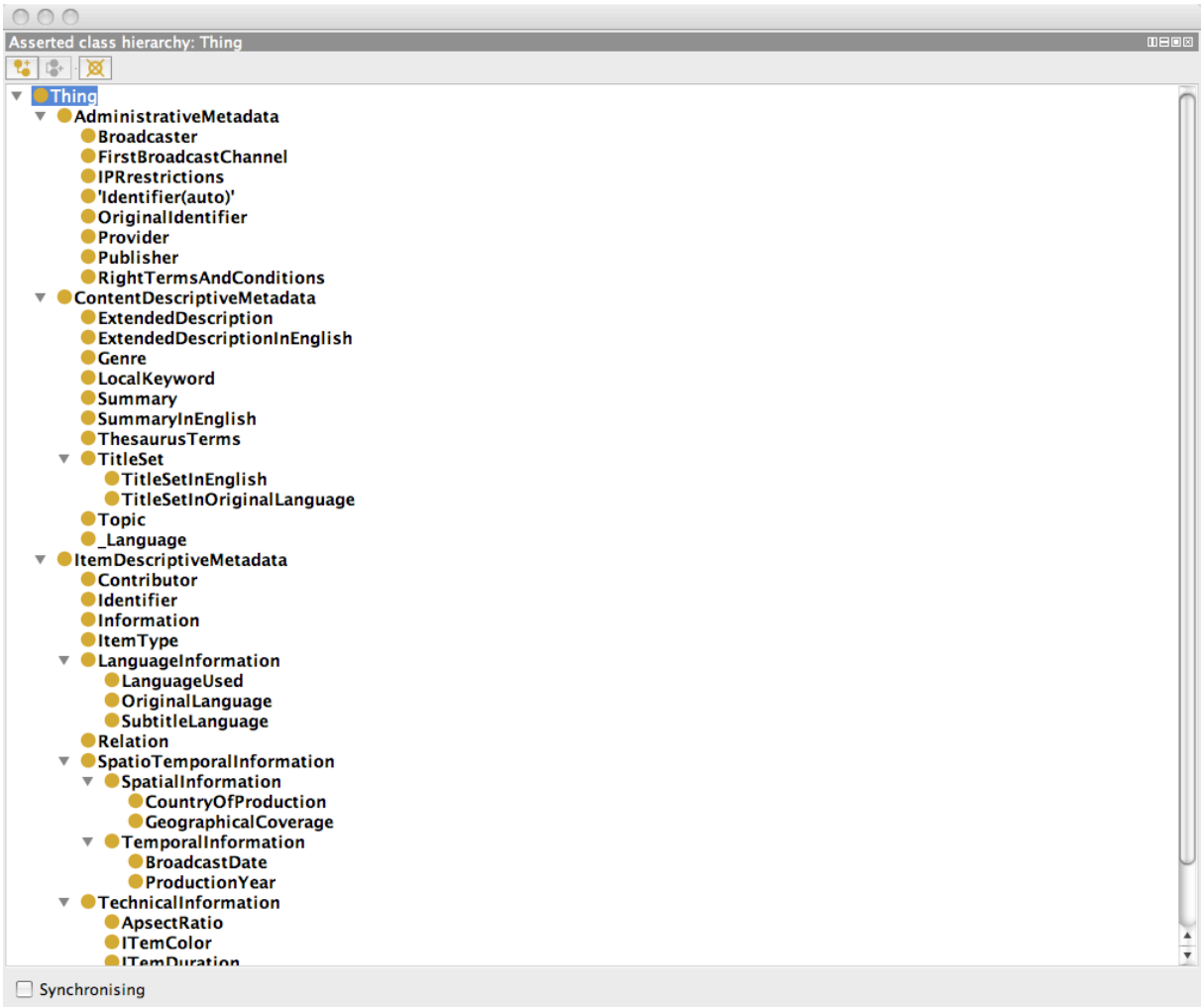

Figure 3.1: The class hierarchy of the EUScreen ontology

The top concept of the ontology is the "thing" (top concept for all OWL ontologies). "Thing" has 3 subconcepts "administrative metadata", "Content descriptive metadata", and "item descriptive metadata". Each of these categories contain a number of subconcepts as can be seen in fig. 3.1.

In the figure below (3.2), the list of datatype properties are presented.

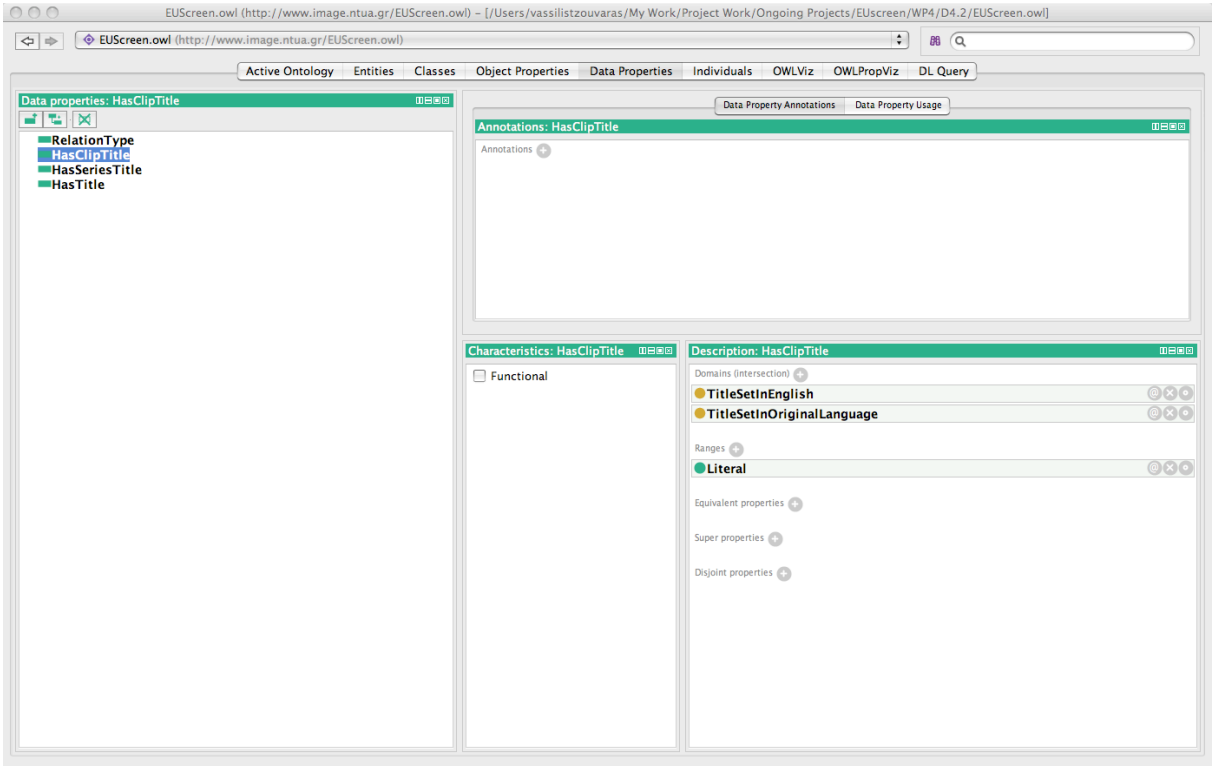

Figure 3.2: Datatype properties

Datatype properties are used to connect concepts with literals and object properties to connect concepts with concepts.

A complete visual presentation of the ontology is presented below (3.3)

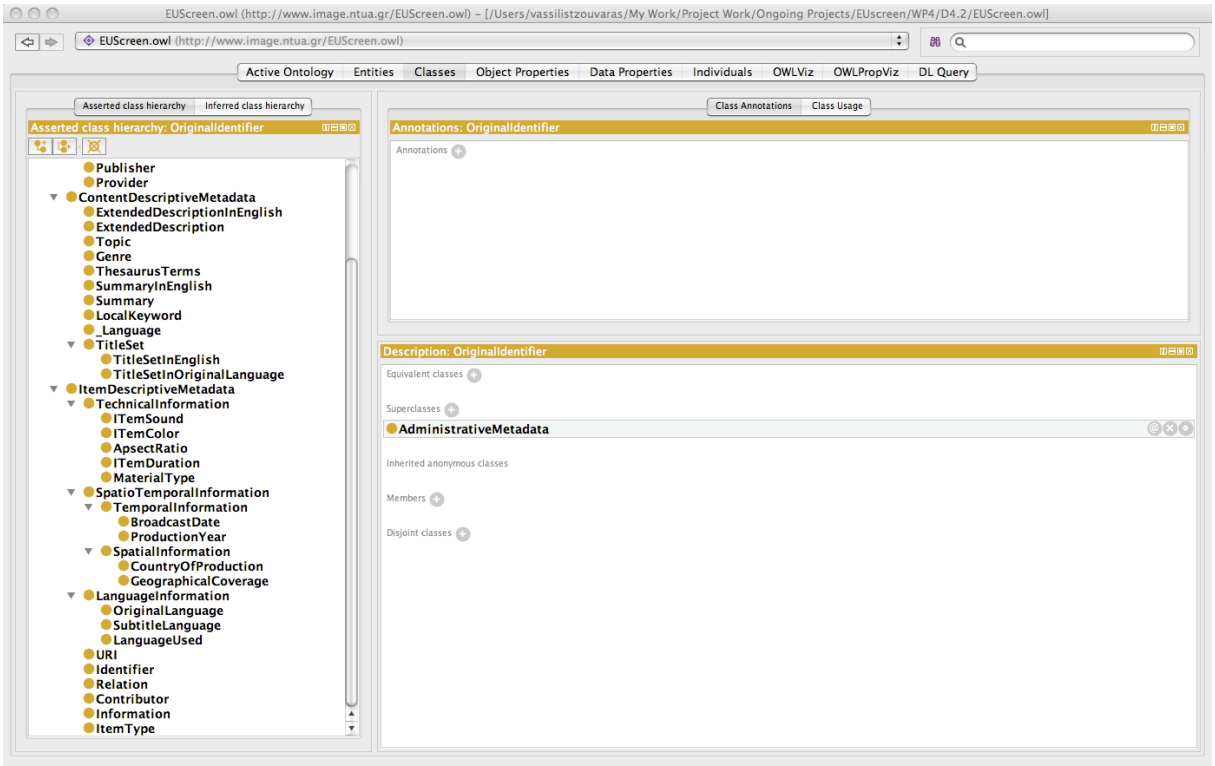

Figure 3.3 Complete view of the ontology

3.2 Complete Representation

Below the complete ontology in RDF/XML format is presented.

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [

   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >

   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >

   <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >

   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >

   <!ENTITY EUScreen "http://www.image.ntua.gr/EUScreen.owl#" >

   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >

]>

<rdf:RDF xmlns="http://www.image.ntua.gr/EUScreen.owl#"

    xml:base="http://www.image.ntua.gr/EUScreen.owl"

    xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"

    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"

    xmlns:EUScreen="http://www.image.ntua.gr/EUScreen.owl#"

    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"

    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

    xmlns:owl="http://www.w3.org/2002/07/owl#">

   <owl:Ontology rdf:about=""/>

   <!--

   ///////////////////////////////////////////////////////////////////////////////////
```

```
    //
    // Data properties
    //
    ///////////////////////////////////////////////////////////////////////////////////
    <!-- http://www.image.ntua.gr/EUScreen.owl#HasClipTitle -->
    <owl:DatatypeProperty rdf:about="#HasClipTitle">
        <rdfs:domain rdf:resource="#TitleSetInEnglish"/>
        <rdfs:domain rdf:resource="#TitleSetInOriginalLanguage"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
    </owl:DatatypeProperty>
    <!-- http://www.image.ntua.gr/EUScreen.owl#HasSeriesTitle -->
<owl:DatatypeProperty rdf:about="#HasSeriesTitle">
        <rdfs:domain rdf:resource="#TitleSetInEnglish"/>
        <rdfs:domain rdf:resource="#TitleSetInOriginalLanguage"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
    </owl:DatatypeProperty>
    <!-- http://www.image.ntua.gr/EUScreen.owl#HasTitle -->
    <owl:DatatypeProperty rdf:about="#HasTitle">
        <rdfs:domain rdf:resource="#TitleSetInEnglish"/>
        <rdfs:domain rdf:resource="#TitleSetInOriginalLanguage"/>
        <rdfs:range rdf:resource="&rdfs;Literal"/>
    </owl:DatatypeProperty>
    <!-- http://www.image.ntua.gr/EUScreen.owl#RelationType -->
```

```
<owl:DatatypeProperty rdf:about="#RelationType">
    <rdfs:domain rdf:resource="#Relation"/>
    <rdfs:range rdf:resource="&rdfs;Literal"/>
</owl:DatatypeProperty>
<!--
///////////////////////////////////////////////////////////////////////////////////
//
// Classes
//
///////////////////////////////////////////////////////////////////////////////////
<!-- http://www.image.ntua.gr/EUScreen.owl#AdministrativeMetadata -->

<owl:Class rdf:about="#AdministrativeMetadata"/
<!-- http://www.image.ntua.gr/EUScreen.owl#ApsectRatio -->
<owl:Class rdf:about="#ApsectRatio">
    <rdfs:subClassOf rdf:resource="#TechnicalInformation"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#BroadcastDate -->
<owl:Class rdf:about="#BroadcastDate">
    <rdfs:subClassOf rdf:resource="#TemporalInformation"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#Broadcaster -->
<owl:Class rdf:about="#Broadcaster">
```

```
    <rdfs:subClassOf rdf:resource="#AdministrativeMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#ContentDescriptiveMetadata -->
<owl:Class rdf:about="#ContentDescriptiveMetadata"/>
<!-- http://www.image.ntua.gr/EUScreen.owl#Contributor -->
<owl:Class rdf:about="#Contributor">
    <rdfs:subClassOf rdf:resource="#ItemDescriptiveMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#CountryOfProduction -->
<owl:Class rdf:about="#CountryOfProduction">
    <rdfs:subClassOf rdf:resource="#SpatialInformation"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#ExtendedDescription -->
<owl:Class rdf:about="#ExtendedDescription">
    <rdfs:subClassOf rdf:resource="#ContentDescriptiveMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#ExtendedDescriptionInEnglish -->
<owl:Class rdf:about="#ExtendedDescriptionInEnglish">
    <rdfs:subClassOf rdf:resource="#ContentDescriptiveMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#FirstBroadcastChannel -->
<owl:Class rdf:about="#FirstBroadcastChannel">
    <rdfs:subClassOf rdf:resource="#AdministrativeMetadata"/>
```

```
</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#Genre -->

<owl:Class rdf:about="#Genre">

    <rdfs:subClassOf rdf:resource="#ContentDescriptiveMetadata"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#GeographicalCoverage -->

<owl:Class rdf:about="#GeographicalCoverage">

    <rdfs:subClassOf rdf:resource="#SpatialInformation"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#IPRrestrictions -->

<owl:Class rdf:about="#IPRrestrictions">

    <rdfs:subClassOf rdf:resource="#AdministrativeMetadata"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#ITemColor -->

<owl:Class rdf:about="#ITemColor">

    <rdfs:subClassOf rdf:resource="#TechnicalInformation"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#ITemDuration -->

<owl:Class rdf:about="#ITemDuration">

    <rdfs:subClassOf rdf:resource="#TechnicalInformation"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#ITemSound -->

<owl:Class rdf:about="#ITemSound">
```

```
    <rdfs:subClassOf rdf:resource="#TechnicalInformation"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#Identifier -->
<owl:Class rdf:about="#Identifier">
    <rdfs:subClassOf rdf:resource="#ItemDescriptiveMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#Identifier(auto) -->
<owl:Class rdf:about="#Identifier(auto)">
    <rdfs:subClassOf rdf:resource="#AdministrativeMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#Information -->
<owl:Class rdf:about="#Information">
    <rdfs:subClassOf rdf:resource="#ItemDescriptiveMetadata"/>
</owl:Class
<!-- http://www.image.ntua.gr/EUScreen.owl#ItemDescriptiveMetadata -->
<owl:Class rdf:about="#ItemDescriptiveMetadata"/>
<!-- http://www.image.ntua.gr/EUScreen.owl#ItemType -->
<owl:Class rdf:about="#ItemType">
    <rdfs:subClassOf rdf:resource="#ItemDescriptiveMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#LanguageInformation -->
<owl:Class rdf:about="#LanguageInformation">
    <rdfs:subClassOf rdf:resource="#ItemDescriptiveMetadata"/>
```

```
</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#LanguageUsed -->

<owl:Class rdf:about="#LanguageUsed">

    <rdfs:subClassOf rdf:resource="#LanguageInformation"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#LocalKeyword -->

<owl:Class rdf:about="#LocalKeyword">

    <rdfs:subClassOf rdf:resource="#ContentDescriptiveMetadata"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#MaterialType -->

<owl:Class rdf:about="#MaterialType">

    <rdfs:subClassOf rdf:resource="#TechnicalInformation"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#OriginalIdentifier -->

<owl:Class rdf:about="#OriginalIdentifier">

    <rdfs:subClassOf rdf:resource="#AdministrativeMetadata"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#OriginalLanguage -->

<owl:Class rdf:about="#OriginalLanguage">

    <rdfs:subClassOf rdf:resource="#LanguageInformation"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#ProductionYear -->

<owl:Class rdf:about="#ProductionYear">
```

```
    <rdfs:subClassOf rdf:resource="#TemporalInformation"/>
</owl:Class
<!-- http://www.image.ntua.gr/EUScreen.owl#Provider -->
<owl:Class rdf:about="#Provider">
    <rdfs:subClassOf rdf:resource="#AdministrativeMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#Publisher -->
<owl:Class rdf:about="#Publisher">
    <rdfs:subClassOf rdf:resource="#AdministrativeMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#Relation -->
<owl:Class rdf:about="#Relation">
    <rdfs:subClassOf rdf:resource="#ItemDescriptiveMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#RightTermsAndConditions -->
<owl:Class rdf:about="#RightTermsAndConditions">
    <rdfs:subClassOf rdf:resource="#AdministrativeMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#SpatialInformation -->
<owl:Class rdf:about="#SpatialInformation">
    <rdfs:subClassOf rdf:resource="#SpatioTemporalInformation"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#SpatioTemporalInformation -->
```

```
<owl:Class rdf:about="#SpatioTemporalInformation">
    <rdfs:subClassOf rdf:resource="#ItemDescriptiveMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#SubtitleLanguage -->
<owl:Class rdf:about="#SubtitleLanguage">
    <rdfs:subClassOf rdf:resource="#LanguageInformation"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#Summary -->
<owl:Class rdf:about="#Summary">
    <rdfs:subClassOf rdf:resource="#ContentDescriptiveMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#SummaryInEnglish -->
<owl:Class rdf:about="#SummaryInEnglish">
    <rdfs:subClassOf rdf:resource="#ContentDescriptiveMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#TechnicalInformation -->
<owl:Class rdf:about="#TechnicalInformation">
    <rdfs:subClassOf rdf:resource="#ItemDescriptiveMetadata"/>
</owl:Class>
<!-- http://www.image.ntua.gr/EUScreen.owl#TemporalInformation -->
<owl:Class rdf:about="#TemporalInformation">
    <rdfs:subClassOf rdf:resource="#SpatioTemporalInformation"/>
</owl:Class>
```

```
<!-- http://www.image.ntua.gr/EUScreen.owl#ThesaurusTerms -->

<owl:Class rdf:about="#ThesaurusTerms">

    <rdfs:subClassOf rdf:resource="#ContentDescriptiveMetadata"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#TitleSet -->

<owl:Class rdf:about="#TitleSet">

    <rdfs:subClassOf rdf:resource="#ContentDescriptiveMetadata"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#TitleSetInEnglish -->

<owl:Class rdf:about="#TitleSetInEnglish">

    <rdfs:subClassOf rdf:resource="#TitleSet"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#TitleSetInOriginalLanguage -->

<owl:Class rdf:about="#TitleSetInOriginalLanguage">

    <rdfs:subClassOf rdf:resource="#TitleSet"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#Topic -->

<owl:Class rdf:about="#Topic">

    <rdfs:subClassOf rdf:resource="#ContentDescriptiveMetadata"/>

</owl:Class>

<!-- http://www.image.ntua.gr/EUScreen.owl#URI -->

<owl:Class rdf:about="#URI">

    <rdfs:subClassOf rdf:resource="#ItemDescriptiveMetadata"/>
```

```
    </owl:Class>

    <!-- http://www.image.ntua.gr/EUScreen.owl#_Language -->

    <owl:Class rdf:about="#_Language">

        <rdfs:subClassOf rdf:resource="#ContentDescriptiveMetadata"/>

    </owl:Class>

</rdf:RDF>

<!-- Generated by the OWL API (version 2.2.1.1101) http://owlapi.sourceforge.net -->
```

4 Conclusion

In this deliverable, the work done for the translation of the EUScreen schema on a semantic web language was presented. The EUScreen ontology will be used for storing, presenting and searching the metadata. The use of semantic web technologies can improve the search functionality and the alignment with external web resources enabling automatic metadata enrichment. The ontology created in the PROTÉGÉ ontology editing tool and has been exported in RDF/XML and OWL formats.

5 References

[1] Description of Work, EUScreen project, http://beeldengeluid.basecamphq.com/clients/1554743

[2] Extensible Markup Language (XML), http://www.w3.org/XML/

[3] Resource Description Framework (RDF) http://www.w3.org/RDF/.

[4] RDF Vocabulary Description Language 1.0: RDF Schema http://www.w3.org/TR/rdf-schema/.

[5] Web Ontology Language (OWL) http://www.w3.org/2004/OWL/.

[6] Borgida, M. Lenzerini, and R. Rossati. The Description Logic Handbook: Theory, Implementation and Applications. Cambridge University Press.

[7] José M. Martínez. MPEG-7 Overview, http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm.

[8] Jan Bormans, Keith Hill. MPEG-21 Overview, http://www.chiariglione.org/mpeg/standards/mpeg-21/mpeg-21.htm.

[9] Dublin Core Metadata Element Set, http://dublincore.org/documents/dcmi-terms/.

[10] SPECTRUM Metadata Standard, http://www.mda.org.uk/spectrum.htm.

[11] Categories for the Description of Works of Art (CDWA) Metadata Standard, http://www.getty.edu/research/conducting_research/standards/cdwa/.

[12] Art Museum Image Consortium (AMICO) Metadata Standard, http://www.amico.org/AMICOlibrary/dataspec.html.

[13] MAchine-Readable Cataloguing 21 (MARC21) Metadata Standard, http://www.bl.uk/services/bibliographic/marc21move.html.

[14] Visual Resources Association (VRA) Core, http://www.vraweb.org/vracore3.htm.

[15] Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH), http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm#Introduction.

[16] IMS Metadata Standard, http://www.imsglobal.org/metadata/index.html.

[17] Alistair Miles, Dan Brickley. SKOS Core Vocabulary Specification, http://www.w3.org/TR/2005/WD-swbp-skos-core-spec-20051102/.

[18] CIDOC-Conceptual Reference Model (CRM), http://cidoc.ics.forth.gr/.

[19] IPTC Core, http://www.iptc.org/IPTC4XMP/.

[20] Lois Mai Chan, and Marcia Lei Zeng. Metadata Interoperability and Standardization – A Study of Methodology, Part I, http://www.dlib.org/dlib/june06/chan/06chan.html

[21] Lois Mai Chan, and Marcia Lei Zeng. Metadata Interoperability and Standardization – A Study of Methodology Part II, http://www.dlib.org/ dlib/june06/zeng/06zeng.html

[22] NISO (National Information Standards Organization). (2004). *Understanding metadata*. Bethesda, MD: NISO Press. http://www.niso.org/standards/resources/UnderstandingMetadata.pdf.

[23] CC:DA (ALCTS/CCS/Committee on Cataloging: Description and Access). (2000). Task Force on Metadata: Final report, June 16, 2000. http://www.libraries.psu.edu/tas/jca/ccda/tf-meta6.html.

[24] A. Taylor. The Organization of Information. 2nd ed. Westport, CN: Libraries Unlimited, 2004.

[25] P. Johnston, Metadata and interoperability in a complex world. Ariadne, 37. http://www.ariadne.ac.uk/issue37/dc-2003-rpt/.

[26] Maria del Mar Roldan-Garcia and Jose F. Aldana-Montes: A Survey on Disk Oriented Querying and Reasoning on the Semantic Web, ICDEW '06: Proceedings of the 22nd International Conference on Data Engineering Workshops (ICDEW'06), 2006, 58.

[27] Broekstra, J., Kampman, A., Harmelen, F. (2002). "Sesame: A Generis Architecture for Storing and Querying RDF and RDF Schema". 1st Internacional Semantic Web Conference (ISWC2002).

[28] RDQL - A Query Language for RDF W3C Member Submission 9 January 2004. http://www.w3.org/Submission/2004/SUBM-RDQL.

[29] The SeRQL query language, http://www.openrdf.org/doc/sesame/users/ch06.html#d0e1056.

[30] B. McBride. Jena: Implementing the RDF Model and Syntax Specification. Steffen Staab et al (eds.): Proceedings of the second international workshop on Semantic Web. SemWeb2001.

[31] K. Wilkinson, C. Sayers, and H. Kuno "Efficient RDF Storage and Retrieval in Jena2", Int. Conf. on Semantic Web and Databases, 2003.

[32] KAON. The Karlsruhe Ontology and Semantic Web Framework. Developer's Guide for KAON 1.2.7. January 2004. http://km.aifb.unikarlsruhe.de/kaon2/Members/rvo/KAON-Dev-Guide.pdf.

[33] Wood, D., Gearon, P., Adams, T. Kowari: A Platform for Semantic Web Storage and Analysis. XTech Conference 2005.

[34] Tucana Technologies, iTQL Commands, http://kowari.org/271.htm.

[35] Pan, Z. and Heflin, J. DLDB: Extending Relational Databases to Support Semantic Web Queries. In Workshop on Practical and Scaleable Semantic Web Systems, ISWC 2003.

[36] Finin, Labrou and Mayfield, A brief introduction to the knowledge interchange format, http://www.cs.umbc.edu/kse/kif/kif101.shtml.

[37] Horrocks, I. The FaCT system. International Conference Tableaux '98, number 1397 in Lecture Notes in Artificial Intelligence. Springer-Verlag, 1998.

[38] Grosof, B.N., Horrocks, I., Volz, R., and Decker, S. Description Logic Programms: Combining Logic Programms with Description Logic. In Proceedings of the 12th International World Wide Web Conference. 2003.

[39] Weithoener, T., Liebig, T., Specht, G. Storing and Querying Ontologies in Logic Databases. The first International Workshop on Semantic Web and Databases. VLDB 2003.

[40] Raghu Ramakrishnan, Divesh Srivastava, S. Sudarshan, and Praveen Seshadri. The CORAL Deductive System. VLDB Journal: Very Large Data Bases, 3(2):161-210, 1994.

[41] Horrocks, I., Li, L., Turi, D., Bechhofer, S. The Instance Store: Description Logic Reasoning with Large Numbers of Individuals. 2004.