AN INTEGRATED ART ANALYSIS AND NAVIGATION ENVIRONMENT

2002-08-13

D6.1 Distributed Query Layer and Metadata Report

Project acronym	ARTISTE			
Contract number	IST 11.978			
Deliverable number	D6.1			
Deliverable title	Distributed Query Layer and Metadata Report			
Workpackage	WP6 Distributed Query and Metadata Layer			
Task	T6.1, T6.2, T6.3			
Date of delivery	**Contractual**	PM24	**Actual**	2002-08-13
Code name	D905-0004654.a		**Version 2.0**	draft ☐ final ☒
Nature	Report			
Dissemination level	IST Programme			
Authors (Partner)	UoS IT Innovation			
Contact Person	Alison Stevenson IT Innovation Centre 2 Venture Road Chilworth Science Park, Southampton S016 7NP, United Kingdom		Tel: +44 23 8076 0834 Fax: +44 23 8076 0833 mailto:as@it-innovation.soton.ac.uk http://www.it-innovation.soton.ac.uk	
Abstract	This report discusses the design and implementation of the distributed query layer of the ARTISTE content-based image retrieval and cataloguing system. The discussion encompasses the techniques used and common standards utilised to enable transparent translation of local metadata schema to common standards and the provision of an open interface for cross-collection search and retrieval that advances open standards for remote access to digital libraries.			

Document Changes

Rev.	Date	Section	Comment
A	2002-08-13	All	Initial Issue

Reviewers of Current Revision

Rev.	Name, Organization	Role
A	ARTISTE PMT	Project Management Team
	Warren Sterling	Project Board Chair

Conventions Used in This Document

The following notational conventions are used in this document:

- Variable and Styles names are shown in Arial 9 pt: Variable1
- Code is shown in Courier New 10 pt: `While True Do`
- Commands are shown in Courier New 11 pt: `Delete`

Trademarks

Control Information

Page 65 is the last page of this document.

Executive Summary

This document is the D6.1 deliverable – Distributed Query Layer and Metadata Report – for the ARTISTE project. ARTISTE is European Commission supported project that has developed integrated content and metadata-based image retrieval across several major art galleries in Europe. Collaborating galleries include the Louvre in Paris, the Victoria and Albert Museum in London, the Uffizi Gallery in Florence and the National Gallery in London.

Museums and galleries often have several digital collections ranging from public access images to specialised scientific images used for conservation purposes. Cross-collection access is recognised as important, for example to compare the treatments and conditions of Europe's paintings, which form a core part of our cultural heritage. However direct access from one gallery to another is currently uncommon for textual data and almost unheard of in terms of image-based search and retrieval. Thus access to a wealth of digital image information is limited.

In part this is because of a lack of relevant metadata to describe the images, in part because that metadata which does exist does not conform to a common schema, and in part exploitation of digital collections is limited because of a lack of appropriate and convenient access methods.

Over the last two and a half years, ARTISTE has developed an image search and retrieval system that integrates distributed, heterogeneous image collections, providing a single interface to the art and its metadata. In doing so ARTISTE has addressed issues of interoperability at the metadata level, at the access protocol level and at the linguistic level.

This report describes

- the innovative use of emerging technical standards such as XML, RDF, along with common metadata standards such as Dublin Core, to enable users to seamlessly execute cross-collection metadata searches while maintaining the diverse legacy database schemas of those collections
- the further use of such standards to enable multilingual access to the metadata which is itself multilingual
- the use of image processing algorithms in combination with XML and RDF standards to allow users to catalogue art objects and dynamically generate metadata describing the images
- the support for and extension of standard information retrieval protocols such as the Open Archive Initiative Metadata Harvesting Protocol and the z39.50 digital library standard to create an open interface for cross-collection search and retrieval

The intended audience for this report includes museum and gallery owners who are interested in providing or extending services for remote access, developers of collection management and image search and retrieval systems.

Contents

1. Introduction

1.1 ARTISTE Project Overview

The ARTISTE project [i], partly funded by the EU under the fifth R&D framework, has developed a system for the automatic indexing and cross-collection search and retrieval of high-resolution art images.

Four major European galleries are involved in the project: the Uffizi in Florence, the National Gallery and the Victoria and Albert Museum in London, and the Centre de Recherche et de Restauration des Musées de France (C2RMF) which is the Louvre related restoration centre. The ARTISTE system currently holds over 160,000 images from four separate collections owned by these partners.

The galleries have joined forces with NCR, a leading player in database and Data Warehouse technology; Interactive Labs, the new media design and development facility of Italy's leading art publishing group, Giunti; IT Innovation, a specialist in building innovative IT systems; and the Department of Electronics and Computer Science at the University of Southampton.

Two and a half years after the project began, and with over twenty person-years of effort by the partners, the ARTISTE project is now reaching its conclusion. An article that provides a summary of the project results has recently been published in Cultivate Interactive [ii].

The work started in ARTISTE is being continued in SCULPTEUR [iii] which is again funded by the European Commission. SCULPTEUR will develop both the technology and the expertise to create, manage and present cultural archives of 3D models and associated multimedia objects.

1.2 ARTISTE System Overview

ARTISTE uses a multi-tier architecture as shown in Figure 1-1.

The architecture consists of software layers separated by a series of APIs (Application Programming Interfaces) that enable loose coupling to be achieved between these main architectural components.

The use of documented and published APIs is necessary in order to establish a common development framework that allow components to be developed by different organisations.

The use of open standards such as OAI and SRW for certain interfaces provides the basis of interoperability with other software systems, for example digital libraries, that haven't been specifically developed to work with ARTISTE.

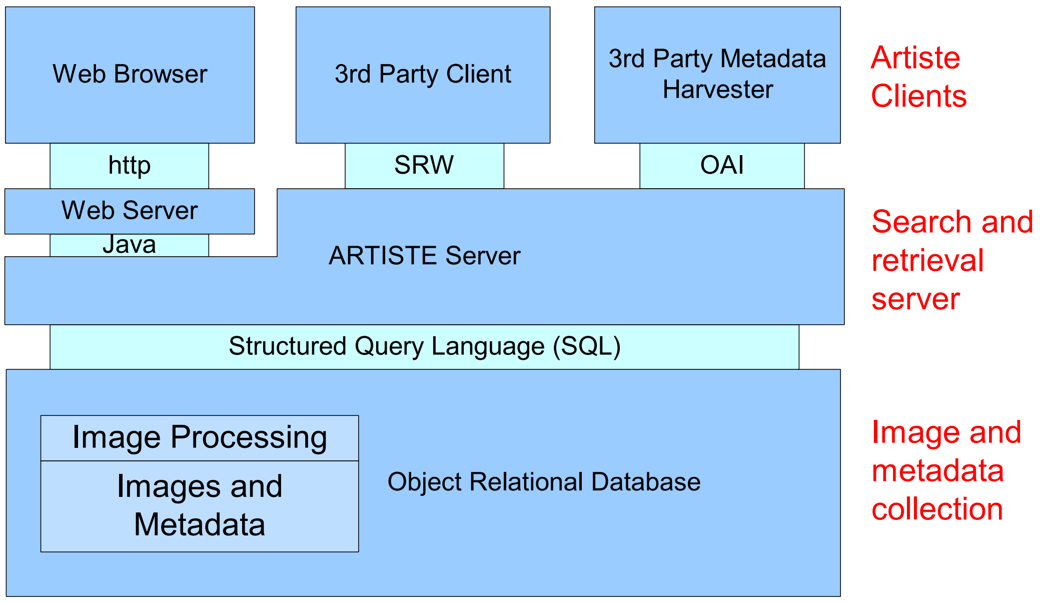

Web Browser
3rd Party Client
3rd Party Metadata Harvester
Artiste Clients
http
SRW
OAI
Web Server
Java
ARTISTE Server
Search and retrieval server
Structured Query Language (SQL)
Image Processing
Images and Metadata
Object Relational Database
Image and metadata collection

Figure 1-1 ARTISTE logical architecture

Images of the art objects in a museum or gallery collection are held in an Object Relational Database Management System (ORDBMS) from NCR called Teradata Object Relational (TOR). Images are stored as Binary Large Objects (BLOBS).

Textual metadata is also stored in the database alongside the images. Storing the images and metadata in the database allows the scalability and robustness of the database management system to be utilised. This is important in light of the size of the image collections involved in the project. For example, one of the collections held in ARTISTE has over 100,000 images, each of which has over 50 different metadata attributes.

ARTISTE uses a wide variety of image processing algorithms as the basis of content-based retrieval. In ARTISTE, each algorithm is applied to the images in the collection to generate a set of image content descriptors called 'feature vectors'. A feature vector can be considered as a way of indexing an image to describe an aspect such as colour distribution or texture. The feature vectors are then integrated and stored with the textual metadata for each image in the database. Creating all the feature vectors in advance for a collection of images greatly improves search time since they do not need to be created every time a content-based search is performed.

A user wishing to perform a content-based search will submit a query image. This image may contain a particular object that they wish to locate in a collection. Alternatively it may contain a particular range of colours, e.g. a certain pigment, The user may be interested finding images with similar colours in the image collection. When a content-based search needs to be made, the required algorithm is run on the query image to create a query feature vector.

The query feature vector is then compared with all the corresponding feature vectors for the images in the collection. The comparison of feature vectors results in a measure of distance between the query image and each image in the collection. The images in the collection are then returned to the user as a series of thumbnails in order of increasing distance. In some cases, the algorithms can be combined into composite queries and a normalised distance measure for each algorithm is used to determine the overall match of a result image to the query. The image processing algorithms are executed in the database by wrapping them as User Defined Modules (UDM) that can be called directly from SQL queries.

The ARTISTE server uses SQL queries to access the textual metadata, feature vectors, images and image processing algorithms in the object relational database. The server is responsible for translating queries from the client applications into SQL that is appropriate for the local metadata and image schema. The ARTISTE server is also able to communicate with other ARTISTE servers so that the query can be executed across multiple, distributed collections. The server is implemented as an Enterprise Java application.

A user can access the ARTISTE server using several different protocols:

- Web based access can be made from a standard browser a ARTISTE supports a simple, wizard driven user interface that makes it easy to create, execute and browse the results of complex metadata and content-based queries.
- Metadata can be extracted from the ARTISTE server through support for the OAI Metadata Harvesting protocol. This only allows the textual metadata on the images to be retrieved.
- Third party software applications can execute metadata and content-based searches through the SRW interface. These applications could be third-party browsers, eLearning applications, other digital library systems, and other ARTISTE servers.

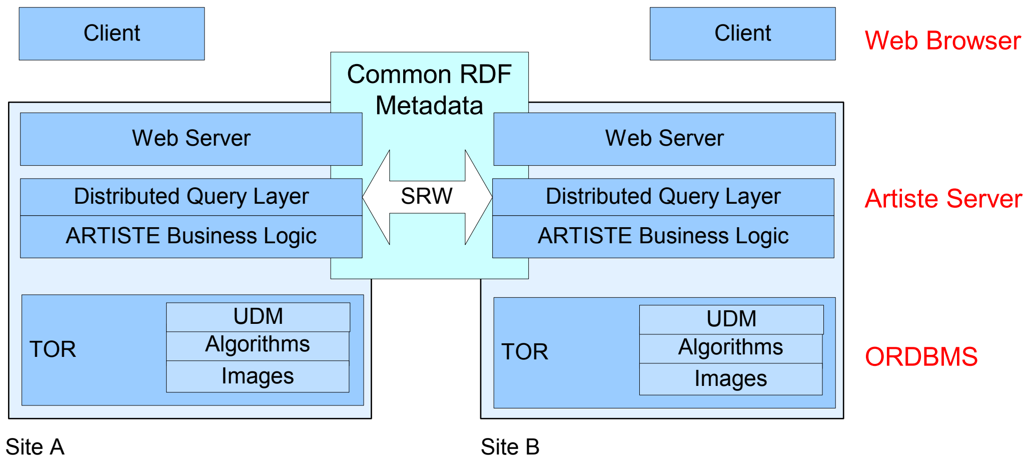


Figure 1-2 ARTISTE physical architecture

The use of a multi-tier architecture allows physical distribution of the system components so that cross-collection searching can be performed when the image collections are physically located at each of the gallery sites. This is achieved by having a local image database and ARTISTE server at each site that owns an image collection.

The distributed architecture gives the galleries control over their own collections and the support for legacy database schemas keeps to a minimum the effort required by galleries make those collections open to retrieval, navigation and interoperability with other collections.

The ARTISTE servers communicate with each other through a ‘distributed query layer’ that uses the SRW (Search and Retrieve Web service) protocol as shown in Figure 1-2. The distributed query and metadata layer provides a single interface to the art, its metadata, and facilities to enable queries to be directed towards multiple distributed databases. SRW provides a framework for search and retrieval but isn’t in itself sufficient to provide interoperability. A common metadata schema is required to define the semantics of the queries made between servers and the data that is returned in response. This common schema is structured using RDF and is mapped to the local metadata schemas of the individual collections.

1.3 Report Structure

The remainder of this report is divided into the following sections.

Section 2 describes in more detail the use of emerging technical standards such as XML and RDF to achieve the transparent translation of local metadata schema to common standards. This section also describes how the use of RDF and RDFS enables ARTISTE to offer multilingual access both to metadata element and to the thesauri which control the values associated with those elements

Section 3 describes dynamic generation of metadata to classify images and the integration of this metadata into the RDF.

Section 4 describes the implementation of support for the Open Archive Initiative Metadata Harvesting Protocol.

Section 5 describes the implementation of the Search and Retrieval Web Service that forms that basis of the ARTISTE Distributed Query Layer. Particular attention is paid to the extensions that were required to be made to current protocol in order to enable both metadata and image content based queries to be executed.

Section 6 presents a brief discussion of the impact on standards made by the ARTISTE project.

Section 7 summarises the findings of this report into a set of conclusions

2. Metadata

2.1 Metadata Storage

As illustrated above in Figure 1-2 each site has a complete instance of the ARTISTE system with its own images and metadata.

Metadata relating to the art objects in a collection is held in an object relational database from NCR (TOR: Teradata Object Relational). More information about the ARTISTE system architecture is available in D8.3 System Integration [iv]. The metadata is a combination of pre-existing data loaded from gallery legacy systems and data generated directly by the ARTISTE system.

The support for legacy database schemas keeps to a minimum the effort required by galleries to make those collections open to retrieval, navigation and interoperability with other collections.

2.2 RDF Mapping

There is no requirement for the pre-existing metadata from the various collections to conform to a single schema. Whereas previous European research projects on Art such as Aquarelle [v] used a standard metadata format to integrate collections, the ARTISTE system maintains the individual database schemas of the galleries. Thus the field 'Ecole' in the C2RMF database will still be called 'Ecole' in the ARTISTE database. ARTISTE enables metadata queries to be executed across the multiple, diverse collections by using Resource Description Format (RDF) [vi] to define the syntax and semantics for standard metadata terms.

Each of the different database schemas employed by the galleries is encoded as an RDF schema, using RDF syntax and the basic semantic expressions of the RDF Vocabulary Description Language 1.0 [vii] more commonly known as RDFS. Each of the four galleries participating in the ARTISTE project use different database schemas to structure their storage of metadata relating to their collections. Each of these database schemas has been encoded using RDF. The four resulting Collection schemas also make reference to the RDF Resources (Classes, Properties and SubClasses) defined in the ARTISTE Core RDF schema, and to the Dublin Core.

2.2.1 ARTISTE Core Schema

The ARTISTE Core schema uses RDF and RDFS to define a query ontology which defines and describes the objects, methods and operators required to build a query based on either image content, textual metadata or a combination of the two. This vastly simplifies specification of complex searches.

A base term for all aspects that can be queried is a Query Item. This can be an image, properties of an image (such as colour or shape), or attributes associated with an image (conventional metadata such as textual and numeric items). This structure, illustrated in Figure 2-1, is defined in the ARTISTE Core schema and reflected in the EJB architecture of the application.

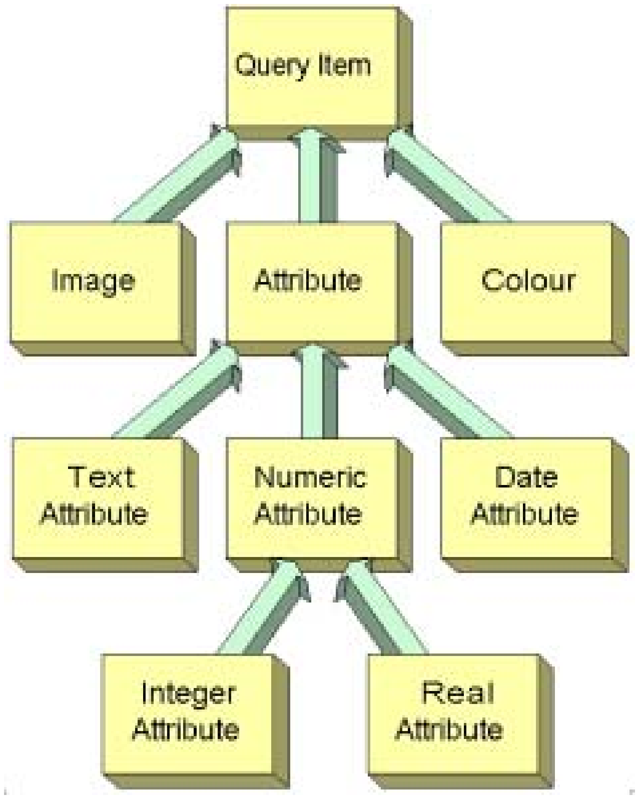


Figure 2-1 *Artiste Query Item hierarchy*

ARTISTE Core further defines a Query Operator as an abstract operation that can be performed on query items. Figure 2-2 shows the current query operators. These include exact operators (such as equals, less than etc.) and fuzzy operators (such as similar to).

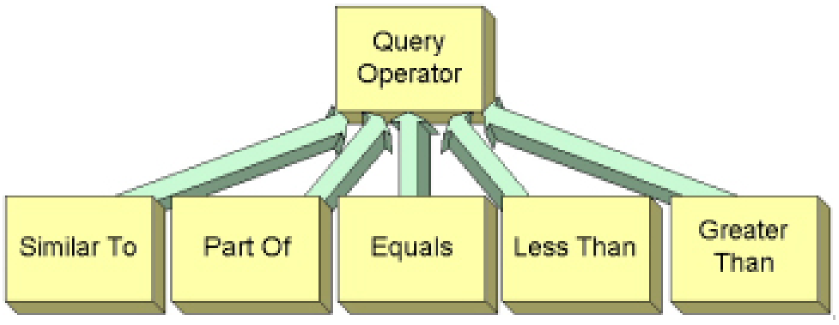


Figure 2-2 Artiste Query Operator hierarchy

Again these are defined in the ARTISTE Core RDF and reflected in the architecture of the java application. An extract showing the declaration of the Query Operator is shown in Figure 2-3

```
<rdfs:Class rdf:ID="QueryOperator">
<rdfs:label xml:lang="en">QueryOperator</rdfs:label>
<rdfs:comment>An operator that can be specified in an ARTISTE
query.</rdfs:comment>
</rdfs:Class>
<rdfs:Class rdf:ID="PartOf">
<rdfs:subClassOf rdf:resource="#QueryOperator"/><rdfs:label
xml:lang="en">PartOf</rdfs:label>
<rdfs:comment>The concept of being part of another
object.</rdfs:comment>
</rdfs:Class>
```

Figure 2-3 Extract from Artiste Core RDF schema defining a QueryOperator Resource

All operators must be instantiated to form rules – each rule describes a legal ARTISTE query expression and specifies which query items can be used with each query operator. Each query operator is assumed to take two operands: a subject and an object. Properties are also defined which govern the particular query items that a particular operator can take.

Figure 2-4 shows a diagrammatic representation of the “SimilarSubImage" query expression rule. This rule defines “SimilarSubImage" as an operation involving two Images and the PartOf operator.

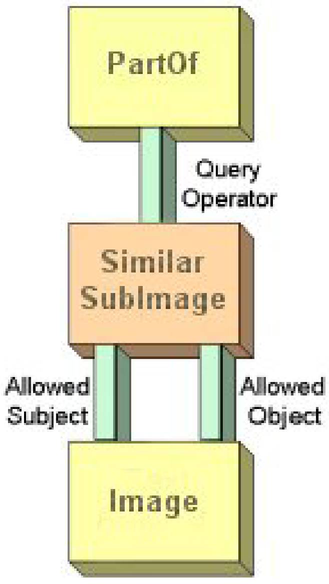


Figure 2-4 SimilarSubImage Query Rule

Figure 2-5 contains the RDF which describes this rule.

```
<ac:QueryExpressionRule rdf:ID="SimilarSubImage">
<ac:QueryOperator>#PartOf</ac:QueryOperator>
<ac:AllowedSubject>#Image</ac:AllowedSubject>
<ac:AllowedObject>#Image</ac:AllowedObject>
<rdfs:label xml:lang="en">Similar SubImage</rdfs:label>
<rdfs:comment>The concept of an image being similar to part of
another image.</rdfs:comment>
</ac:QueryExpressionRule>
```

Figure 2-5 Extract from ARTISTE Core RDF Schema defininng a QueryExpressionRule for SimilarSubImages

Rules governing non-fuzzy operators such as DateEquals or TextContains, can be defined simply as illustrated in Figure 2-5. However rules that contain fuzzy operators, such as PartOf must be further qualified by relating to an analyser.

As described in the System Overview (Section 1.2) ARTISTE defines an algorithm as being a software module that operates on an image to generate or compare image feature vectors. An analyser is an algorithm together with the relevant metadata which describes how the algorithm interacts with the rest of the system. Analysers may be used in conjunction with fuzzy operators (e.g. similar to) or may be used to generate conventional metadata terms.

The analysers which calculate the feature vectors must be applied to query expression rules for them to be used by the system. This is achieved by using the ARTISTE Core defined RDFS property AnalyserAppliesTo to map the analysers to the query expression rules to which they apply. For example it may be used to say that the MultiScalarColour Histogram analyser is used to find similar sub images as illustrated in Figure 2-6 and encoded in RDF in Figure 2-7.

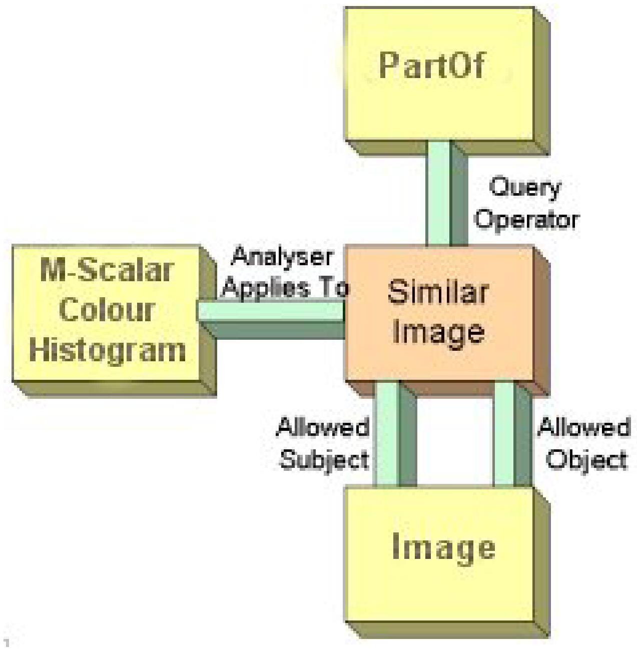


Figure 2-6

```
<ac:Analyser rdf:ID="MultiScalarColourHistogram">

<rdfs:label xml:lang="en">en:MultiScalarColourHistogram</rdfs:label>

<rdfs:comment>Multi Scalar Colour Content</rdfs:comment>

<ac:AnalyserAppliesTo>http://artiste.it-
innovation.soton.ac.uk/rdf/ARTISTECore.rdf#SimilarSubImage</ac:AnalyserAppliesTo>

</ac:Analyser>
```

Figure 2-7

The full ARTISTE Core schema, which has been validated according to the RDF Model and Syntax Specification[vi] using the W3C RDF Validation Service *[**Error! Bookmark not defined.**]*, is available expressed as XML in Appendix A. The schema is also available on the Web at http://artiste.it-innovation.soton.ac.uk/rdf/ArtisteCore.rdf

The Collection schemas use the common syntax of RDF and the vocabulary defined and described in ARTISTE Core to encode the legacy database schemas employed by the various galleries and museum in a machine-readable format.

The Collection schemas also include labels and comments.

Furthermore, where appropriate the Collection schemas use the RDFS utility property 'isDefinedBy' (a subproperty of rdfs:seeAlso) to map the legacy database tables and fields to common metadata standards such as that defined by the Dublin Core Metadata Initiative. This use of the rdfs:isDefinedBy property is in line with the W3C recommendation that the property be used to indicate the resource defining the subject resource.

2.2.2 Dublin Core Schema

The Dublin Core Metadata Initiative (DCMI) have defined a metadata element set consisting of 15 elements [viii].

Each Dublin Core element is defined by the DCMI using a set of ten attributes from the ISO/IEC 11179 standard for the description of data elements. While there is no official DCMI RDF/RDFS encoding of the element set, the DCMI have published a RDF Schema declaration [ix]. However this schema does not include all the information required by the ARTISTE system, for example the DCMI published schema contains no multilingual information. ARTISTE has therefore produced an RDF/RDFS encoding of the 15 Dublin Core metadata elements which references and extends the DCMI published schema. This is in keeping with the incremental extensibility build into RDF and is the same approach taken by the W3C in its experiments with using RDF to describe images [x].

The ARTISTE RDF encoding of the Dublin Core Metadata Element Set, which has been validated according to the RDF Model and Syntax Specification[vi] using the W3C RDF Validation Service [**Error! Bookmark not defined.**], is available expressed as XML in Appendix B. The schema has also been made available on the Web at http://artiste.it-innovation.soton.ac.uk/rdf/dc.rdf

By referencing the ARTISTE Dublin Core schema the collection schemas provide a mapping that relates standard metadata terms to individual database table and column values. Thus the legacy metadata provided by the galleries and museums stored using diverse legacy database schemas can be seamlessly accessed using ARTISTE. Queries are composed using RDF, and subsequently translated to SQL at each site (Figure 2-8).

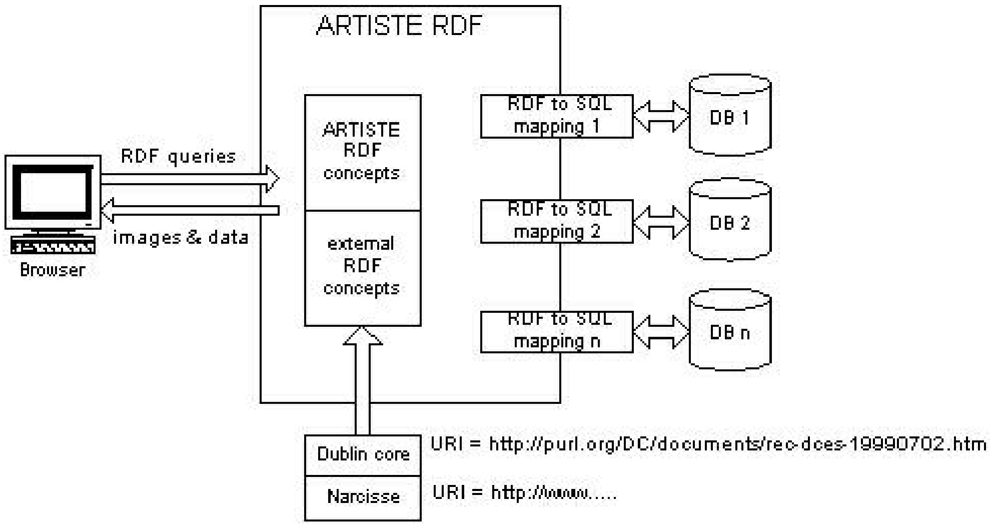


Figure 2-8 Use of RDF in the ARTISTE system

In the example shown in Figure 2-9 interoperability between disparate collections is achieved by mapping the Dublin Core term title to columns in metadata tables. Collection A defines image title within the Caption column and Collection B defines image title within the TitleM column.

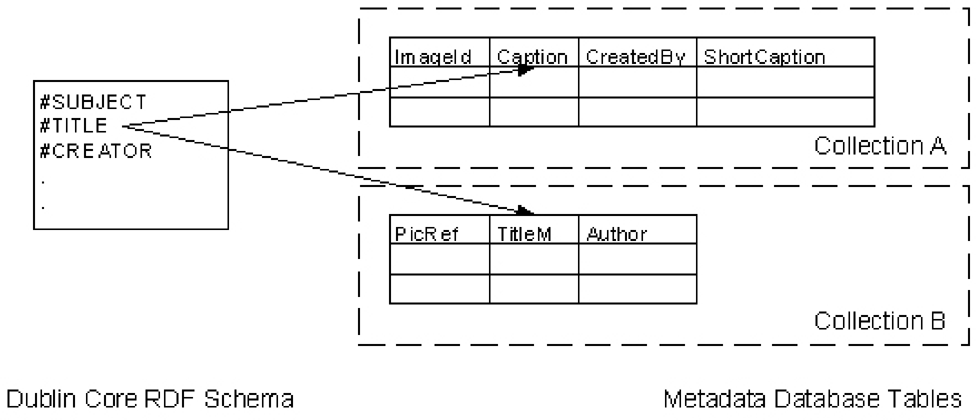


Figure 2-9 Application of Dublin Core in ARTISTE

An extract from the RDF for Collection A (the Victoria and Albert Museum Collection) is shown in Figure 2-10.

```
<ac:TextAttribute rdf:ID="Caption">
  <rdfs:isDefinedBy rdf:resource="http://artiste.it-
    innovation.soton.ac.uk/rdf/vam/vam.rdf.rdf#Caption"/>
  <rdfs:isDefinedBy rdf:resource="‟ http://artiste.it-
    innovation.soton.ac.uk/rdf/dc.rdf.rdf##title”/>
</ac:TextAttribute>
```

Figure 2-10 Extract from the Collection schema for the Victoria and Albert Museum creating a mapping of Dublin Core 'Title'

An extract from the RDF for Collection B (the C2RMF Collection) would look very similar with the TextAttribute TitleM being defined both by itself (for the sake of completeness) and by the ARTISTE Dublin Core schema already published on the web.

In this case a user can then execute a single query across both collections and at each collection the RDF URI for DC Title is translated to the SQL for selecting Caption or TitleM.

The translation to SQL is performed by the ARTISTE application server using the SiRPAC RDF Parser [xi]

As it is the responsibility of each site to define which metadata terms are supported, it is unlikely that collections will provide the same querying capabilities. For example, each collection only supports a subset of the Dublin Core and/or additional terms defined in other standard metadata schemas. To ensure interoperability the ARTISTE system generates a Query Context when a user builds a new query based upon the collections they wish to search. The Query Context, which is constructed from the RDF parsed by SiRPAC, contains a intersection of query capabilities supported by the selected collections.

2.3 Multilingual Metadata

Interoperability between collections in a European context is not a purely technical problem. Any system seeking to provide seamless access to multiple geographically and culturally distributed collections must take account of the multilingual nature both of the metadata and the requirements of users who wishes to access that metadata.

The ARTISTE system contains metadata in English (from the Victoria & Albert Museum and the National Gallery), French (from the C2RMF) and Italian (from the Uffizi). Furthermore the web interface to ARTISTE supports four languages (English, French, Italian and Danish) reflecting the language needs of the members of the consortium.

ARTISTE provides a technical solution to the latter part of this linguistic challenge by including multiple labels and comments for each query item defined in the Collection schemas and uses the XML special attribute ‘xml:lang’ to specify the language used in the content of the element. The values of the attribute are language identifiers as defined by the IETF Standard RFC 1766, Tags for the Identification of Languages [xii].

However the SiRPAC RDF Parser does not support the ‘xml:lang’attribute. Therefore the two letter IEFT language identifier is repeated in the element value string, separated from the actual attribute value by a semicolon. The ‘xml:lang’ attribute is retained in the RDF since other RDF Parsers (and potentially future versions of SirPAC) support the attribute and for the ARTISTE RDF schemas to be useful beyond the life of the project it is desirable that they conform to best practise. An example taken from the C2RMF collection schema is shown in Figure 2-11

```
<ac:TextAttribute rdf:ID="lauthor">
      <rdfs:label xml:lang="en">en:Related
        Artists</rdfs:label>
      <rdfs:label xml:lang="fr">fr:Lien avec
        l'auteur</rdfs:label>
      <rdfs:label xml:lang="it">it:Vinculo con el
        artista</rdfs:label>
      <rdfs:label xml:lang="dk">dk:Relation til
        artisten</rdfs:label>
      <rdfs:comment xml:lang="en">en:Other artists who
        influenced the work of art eg the school of
        GIOTTO DI BONDONE</rdfs:comment>
      <rdfs:comment xml:lang="fr">fr:nom d'autres
        artistes qui on influencela realisation de
        l'oeuvre d'art ex: l'ecole de GIOTTO DI
        BONDONE</rdfs:comment>
      <rdfs:comment xml:lang="it">it:Altri artisti che
        influenzarono l'opera d'arte, per esempio la
        scuola di GIOTTO di BONDONE</rdfs:comment>
      <rdfs:comment xml:lang="dk">dk:Andre kunstnere som
        har pavirket kunstarten f.eks. the school og
        Giotto Di Bondone</rdfs:comment>
      <rdfs:isDefinedBy rdf:resource="http://artiste.it-
        innovation.soton.ac.uk/test_rdf/c2rmf/c2rmf.rdf#l
        author" />
  </ac:TextAttribute>
```

Figure 2-11 Use of RDF to provide multilingual descriptions

The Query Context is generated according to the language Locale selected by the user so that when the RDF schemas are parsed the appropriate language labels and explanatory comments are extracted, stored in the Query Context and displayed to the user.

Therefore the ARTISTE system enables users to specify queries in multiple languages. For example, using the example above, one English speaking user might wish to search the C2RMF for works of art influenced by Giotto do Bondone and would therefore submit a query where “Related ARTISTE contains ‘de Bondone’”. A Italian user wishing to perform the same search would submit a query where ‘Vinculo con el artista contains ‘de Bondone’”. In each case the same SQL query is executed against the database and the same results returned to the user.

However this does not address the issue of multilingual attribute values stored in the database. For example a user might wish to search across the National Gallery, the Uffizi and the C2RMF collections for images called “The Forest”. The RDF mapping between the legacy database schemas enables the user to execute a single query across all three sites, by searching the those fields mapped to the Dublin Core term Title. However since the metadata stored in the C2RMF database is written in French and the metadata in the Uffizi database is written in Italian, there will be no records in those databases containing the text string ‘Forest’ (instead those databases would contain ‘Forêt’ and ‘Foresta’ respectively).

It is unreasonable to consider translating all the metadata from the various galleries into a common language mainly because this would negate one of the key advantages of the ARTISTE system in that it does *not* require changes to be made to legacy metadata. Furthermore such action would not take account of the fact that users would continue to submit queries containing value strings in various languages which would continue to produce a mismatch with the metadata in one or more of the databases.

A possible solution would be to maintain the legacy metadata in its original languages and perform multiple translations on the query value string supplied by the user, according to which databases the query was being executed over. For example a user, accessing the ARTISTE system in English, might submit a search for images in the Victoria & Albert Museum, the Uffizi Gallery and the C2RMF collection with a title containing the word 'Forest'. The ARTISTE system would build and execute a query for title contains 'Forest' against the Victoria & Albert Museum and corresponding queries for title contains 'Foresta' and title contains 'Foret' against the Uffizi and C2RMF collections respectively. A demonstration BabelFish Web Service [xiii] exists which provides an interface to AltaVista's Babelfish translation application and could be invoked to perform this action. However the service has not been tested and the functionality to make use of real-time translation has not been implemented in ARTISTE.

2.4 Thesauri

In addition to mapping legacy database schemas to common metadata standards (Section 2.2) and enabling multilingual functionality (Section 2.3) ARTISTE uses RDF and RDFS to support metadata thesauri.

Building on the CERED/NBII draft RDF Server Standard Documentation [xiv] ARTISTE supports a total of 15 thesauri for legacy gallery metadata attributes: seven from the Victoria & Albert Museum and eight from the C2RMF. ARTISTE also supports one thesaurus for a dynamically generated metadata element (see Section 3). The thesauri can be broken down into three basic types:

- Simple Controlled Vocabularies
- Controlled Vocabularies using Codes
- Multilingual Controlled Vocabularies

2.4.1 Simple Controlled Vocabularies

For simple controlled vocabularies the list of allowed terms for a given attribute is encoded in RDF using the resources defined in the CERED/NBII schema. This RDF document, or Collection Thesaurus is included by reference in the Collection schema for the gallery collection.

2.4.2 Controlled Vocabularies using Codes

For Controlled Vocabularies using Codes the Collection Thesaurus document contains a list of allowed codes for a given attribute, encoded in RDF using the resources defined in the CERED/NBII schema. The values these codes refer to are stored in a separate RDF document, again encoded using the resources defined in the CERED/NBII schema. This single attribute thesaurus is included by reference in the Collection Thesaurus which is in turn included by reference in the Collection schema for the gallery collection.

2.4.3 Multilingual Controlled Vocabularies

Multilingual functionality was highlighted above as an important part of achieving interoperability (Section 2.3). Multilingual Controlled Vocabularies provide such functionality at the level of the thesaurus and are simply a special case of Controlled Vocabularies using Codes. Multilingual vocabularies consist of a list of allowed attribute values (this may be a code or the actual value in a default language representation) encoded in the collection thesaurus document. The values referred to are stored in a separate RDF document utilising the RDFS attribute 'rdfs:label' and the XML attribute 'xml:lang' as well as the using the resources defined in the CERED/NBII schema.

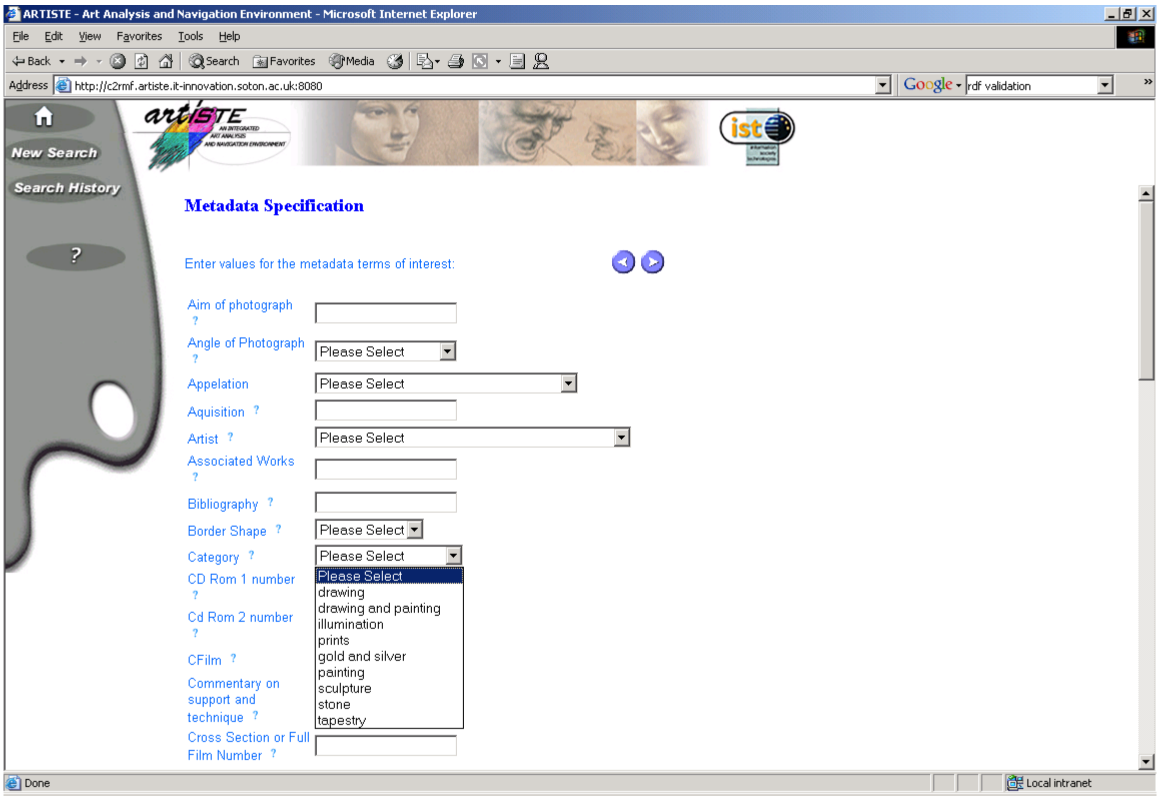
All eight of the thesauri for the C2RMF metadata are multilingual, supporting English and French representations. An example from the Collection Thesaurus for C2RMF and the multilingual controlled vocabulary for the C2RMF metadata attribute Category are shown in Figure 2-13 and Figure 2-12. Figure 2-14 and Figure 2-15 show the resulting user interface. Queries constructed using either interface to the metadata would return the same results.

```
<?xml version="1.0" ?>
- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
    syntax-ns#" xmlns:rdfs="http://www.w3.org/TR/rdf-schema#"
    xmlns:ac="http://artiste.it-
    innovation.soton.ac.uk/rdf/ARTISTECore.rdf#"
    xmlns:aconf="http://artiste.it-
    innovation.soton.ac.uk/rdf/ARTISTEConfiguration.rdf#"
    xmlns:z19="http://artiste.it-
    innovation.soton.ac.uk/rdf/thesaurus.rdf#"
    xmlns:categ="http://artiste.it-
    innovation.soton.ac.uk/rdf/c2rmf/c2rmf-
    thesaurus/categ.rdf#">
  <z19:Category rdf:ID="categ">
      <z19:IC>DE</z19:IC>
      <z19:IC>DP</z19:IC>
      <z19:IC>EN</z19:IC>
      <z19:IC>ES</z19:IC>
      <z19:IC>OR</z19:IC>
      <z19:IC>PE</z19:IC>
      <z19:IC>SC</z19:IC>
      <z19:IC>ST</z19:IC>
      <z19:IC>TA</z19:IC>
  </z19:Category>
```

Figure 2-12 Extract from the Collection Thesaurus for the C2RMF Collection listing the codes for the controlled list of allowed 'Category' values

```
<?xml version="1.0" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
    ns#" xmlns:rdfs="http://www.w3.org/TR/rdf-schema#"
    xmlns:ac="http://artiste.it-
    innovation.soton.ac.uk/rdf/ARTISTECore.rdf#"
    xmlns:aconf="http://artiste.it-
    innovation.soton.ac.uk/rdf/ARTISTEConfiguration.rdf#"
    xmlns:z19="http://artiste.it-
    innovation.soton.ac.uk/rdf/thesaurus.rdf#">
  <z19:Category rdf:ID="DE">
      <rdfs:label xml:lang="en">en:drawing</rdfs:label>
      <rdfs:label xml:lang="fr">fr:dessin</rdfs:label>
    </z19:Category>
  <z19:Category rdf:ID="DP">
      <rdfs:label xml:lang="en">en:drawing and
        painting</rdfs:label>
      <rdfs:label xml:lang="fr">fr:dessin et
        peinture</rdfs:label>
    </z19:Category>
  <z19:Category rdf:ID="EN">
      <rdfs:label
        xml:lang="en">en:illumination</rdfs:label>
      <rdfs:label
        xml:lang="fr">fr:enluminure</rdfs:label>
  </z19:Category>
```

Figure 2-13Extract from the single attribute thesaurus for the C2RMF metadata attribute 'Category'. The thesaurus specifies the multilingual labels for the codes declared in the Collection Thesaurus


ARTISTE - Art Analysis and Navigation Environment - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
New Search
Search History
?
Done
Local intranet

Address: http://c2rmf.artiste.it-innovation.soton.ac.uk:8080

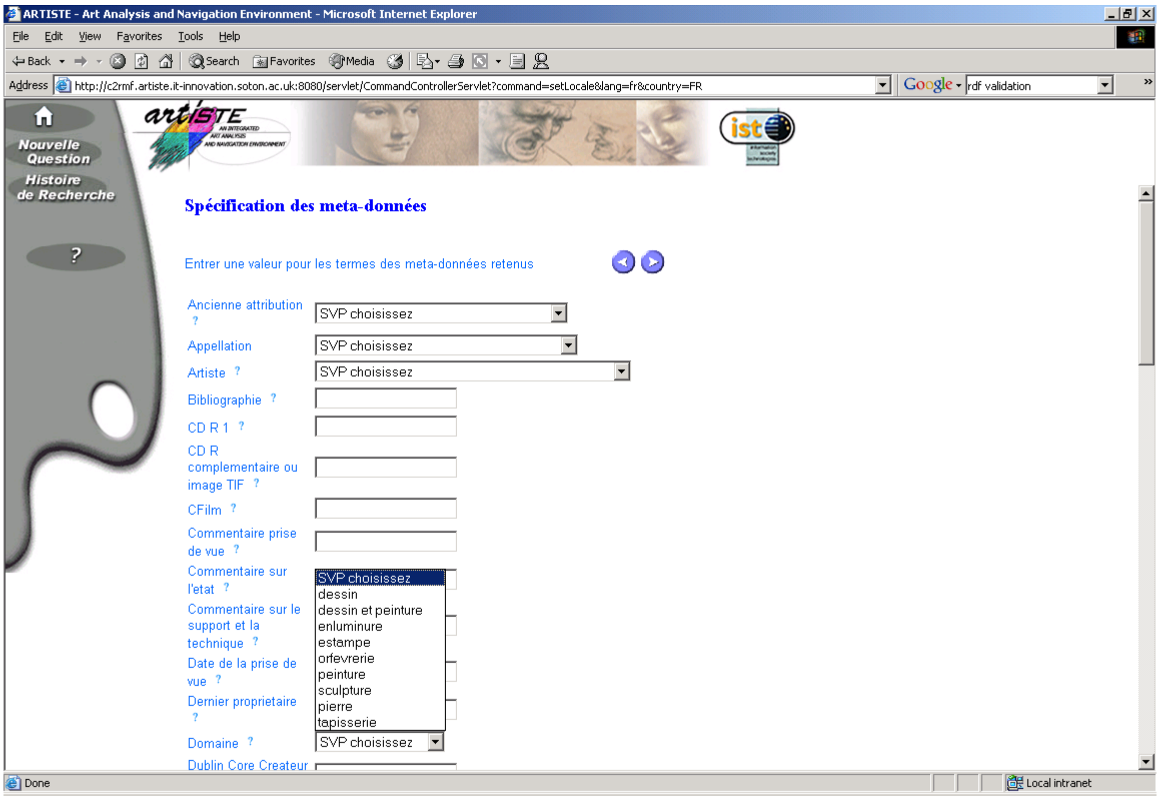
Metadata Specification

Enter values for the metadata terms of interest:

Aim of photograph ?	
Angle of Photograph ?	Please Select
Appelation	Please Select
Aquisition ?	
Artist ?	Please Select
Associated Works ?	
Bibliography ?	
Border Shape ?	Please Select
Category ?	Please Select
CD Rom 1 number ?	
Cd Rom 2 number ?	
CFilm ?	
Commentary on support and technique ?	
Cross Section or Full Film Number ?	

Category list: Please Select, drawing, drawing and painting, illumination, prints, gold and silver, painting, sculpture, stone, tapestry

Figure 2-14 The English language version of the 'Category' thesaurus.


ARTISTE - Art Analysis and Navigation Environment - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Nouvelle Question
Histoire de Recherche
?
Done
Local intranet

Address: http://c2rmf.artiste.it-innovation.soton.ac.uk:8080/servlet/CommandControllerServlet?command=setLocale&lang=fr&country=FR

Spécification des meta-données

Entrer une valeur pour les termes des meta-données retenus

Ancienne attribution ?	SVP choisissez
Appellation	SVP choisissez
Artiste ?	SVP choisissez
Bibliographie ?	
CD R 1 ?	
CD R complementaire ou image TIF ?	
CFilm ?	
Commentaire prise de vue ?	
Commentaire sur l'etat ?	SVP choisissez
Commentaire sur le support et la technique ?	
Date de la prise de vue ?	
Dernier proprietaire ?	
Domaine ?	SVP choisissez
Dublin Core Createur	

Liste: SVP choisissez, dessin, dessin et peinture, enluminure, estampe, orfevrerie, peinture, sculpture, pierre, tapisserie

Figure 2-15 The French language version of the 'Category' thesaurus.

2.4.4 Issues and Solutions

In some cases the legacy metadata attributes controlled by thesauri are the same attributes mapped to common metadata standards such as Dublin Core. This is problematic because although the two (or three, or four) legacy metadata attributes from the various collections are close enough in concept and content to be usefully abstracted to a single standard metadata element it is not the case that the controlled list provided by one gallery for their metadata attribute matches the values contained in the corresponding field in another gallery's database. It is therefore important to associate thesauri with distinct query items in the application server rather than the TextAttributes defined in the collection schemas. This is achieved in the ARTISTE application server which takes the RDF parsed by the SiRPAC and generates a distinct query item in the Query Context.

A second issue which arose in the design and implementation of the multilingual controlled vocabularies was the size of the RDF documents. In some cases the number of allowed terms, each with both an English and a French label produced a document several thousand lines long when encoded in RDF. This proved too large for the SiRPAC parser to process and caused the ARTISTE system to hang. Therefore the contents of the thesaurus was stored inside the database. Although this marks a limitation of the SiRPAC parser it has the added benefit of enabling 'contains' queries across the controlled values.

The Collection Thesaurus RDF documents (c2rmf_thesaurus.rdf and vam_thesaurus.rdf), and a sample single attribute thesaurus RDF document showing multilingual labels (category.rdf), which have been validated according to the RDF Model and Syntax Specification[vi] using the W3C RDF Validation Service [**Error! Bookmark not defined.**], have been published on the Web. The URLs to the thesauri as follows:

Collection Thesaurus for the C2RMF Collection

http://artiste.it-innovation.soton.ac.uk/rdf/c2rmf/c2rmf-thesaurus.rdf

Collection Thesaurus for the Victoria and Albert Collection

http://artiste.it-innovation.soton.ac.uk/rdf/vam/vam-thesaurus.rdf

The single attribute thesauri are available in the following web accessible directories:

Single Attribute Thesauri for the C2RMF Collection

http://artiste.it-innovation.soton.ac.uk/rdf/c2rmf/c2rmf-thesaurus/

Single Attribute Thesauri for the Victoria and Albert Collection

http://artiste.it-innovation.soton.ac.uk/rdf/vam/vam-thesaurus/

3. Dynamic Generation of Metadata

3.1 Automatic Classification

As described in Section 1.2 ARTISTE uses image processing algorithms as the basis of content-based retrieval. While the majority of the algorithms developed in ARTISTE produce feature vectors which are stored in the database and then compared against query feature vectors to produce results some algorithms act as image classifiers.

One such classifier, developed by the IAM group at the University of Southampton, is the border finder algorithm which enables users to classify and retrieve picture frames by their shape, e.g. diamond, circle, ellipse, tondo, triptych, square, rectangle.

There are two parts to this classifier:

- Locating the borders of the picture frame, and
- Labelling the located border with a classification.

For a given image, locating the borders is performed by converging a series of 'sensors' around the edge of the image to the centre of the image. The 'sensors' will stop when they meet a strong change in intensity, which is likely to be due to the presence of a picture frame. The final positions of the sensors indicate the overall shape of the border.

Classifying the border involves a neural network which is trained to recognize shapes. The positions of the sensors are fed into the neural network which then determines the overall shape of the border. The classification process is illustrated in Figure 3-1

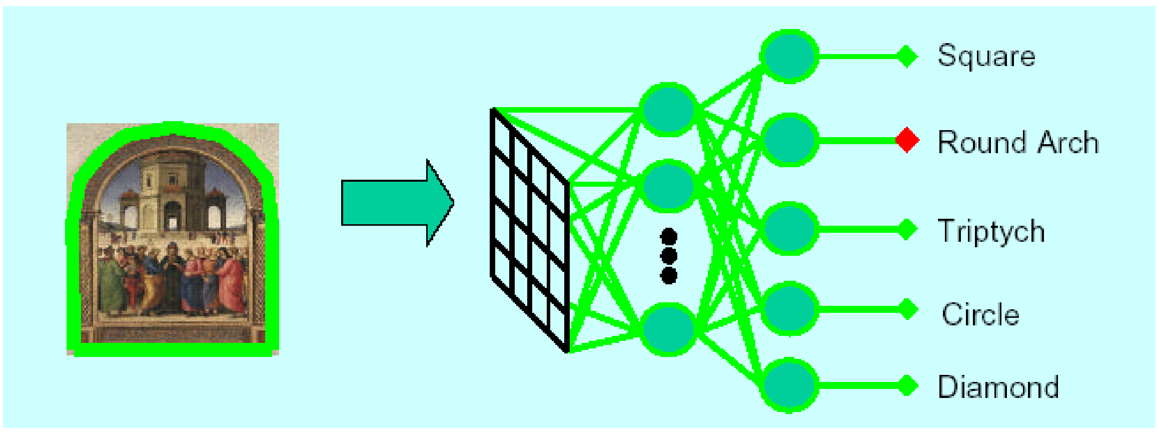


Figure 3-1

The metadata classification produced by the neural net is then inserted automatically into the database.

3.2 Integration through RDF

One of the advantages ARTISTE gains by using RDF to encode information about the metadata formats and elements supported by any one site is that the interface to the system is flexible enough to cope with the dynamic generation of metadata. All that is required for the newly generated metadata to become accessible to users querying the system is that a collection whose images have been classified declares so in the RDF schema describing their metadata. An example of such a declaration from the National Gallery Collection schema is shown in Figure 3-2.

```
<?xml version="1.0" ?>
- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-
    rdf-syntax-ns#"
    xmlns:rdfs="http://www.w3.org/TR/rdf-schema#"
    xmlns:ac="http://artiste.it-
    innovation.soton.ac.uk/rdf/ARTISTECore.rdf#"
    xmlns:aconf="http://artiste.it-
    innovation.soton.ac.uk/rdf/ARTISTEConfiguration.rd
    f#" xmlns:dc="http://artiste.it-
    innovation.soton.ac.uk/rdf/dc.rdf#"
    xmlns:agm="http://artiste.it-
    innovation.soton.ac.uk/rdf/ARTISTEGenMetadata.rdf#
    ">
  :
  <ac:TextAttribute rdf:ID="BorderShape">
      <rdfs:isDefinedBy rdf:resource=
    "http://artiste.itinnovation.soton.ac.uk/rdf/A
        RTISTEGenMetadata.rdf#BorderShape"/>
    </ac:TextAttribute>
```

Figure 3-2

3.2.1 ARTISTE General Metadata Schema

That declaration references an ARTISTE General Metadata Schema which contains the multilingual label information for the user interface. The RDF for the BorderShape metadata element as declared in the ARTISTE General Metadata Schema is shown in Figure 3-3

```
<?xml version="1.0" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
    ns#" xmlns:rdfs="http://www.w3.org/TR/rdf-schema#"
    xmlns:ac="http://artiste.it-
    innovation.soton.ac.uk/rdf/ARTISTECore.rdf#">
  <ac:TextAttribute rdf:ID="BorderShape">
      <rdfs:label xml:lang="en">en:Border
        Shape</rdfs:label>
      <rdfs:label xml:lang="fr">fr:Forme de
        cadre</rdfs:label>
    </ac:TextAttribute>
  </rdf:RDF>
```

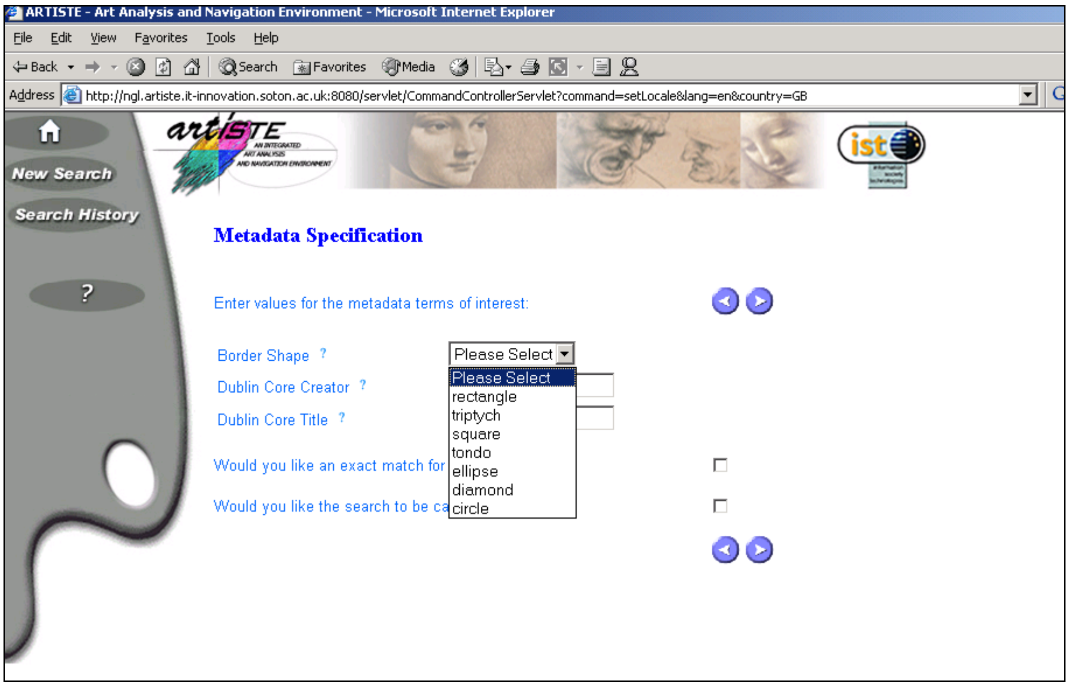
Figure 3-3ARTISTE General Metadata Schema contain TextAttribute description of BorderShape metadata element used to reference dynamically generated metadata.

3.2.2 ARTISTE General Metadata Thesaurus

In the case of the BorderShape metadata element, indeed in most classification scenarios, the metadata values generated by the neural net form a finite set. They can therefore be controlled by a thesaurus which can offer multilingual access to the data. Figure 3-4 shows an extract from the BorderShape RDF thesaurus while Figure 3-5 and Figure 3-6 show the resulting multilingual access to the dynamically generated metadata..

```
<?xml version="1.0" ?>
- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
    syntax-ns#" xmlns:rdfs="http://www.w3.org/TR/rdf-schema#"
    xmlns:ac="http://artiste.it-
    innovation.soton.ac.uk/rdf/ARTISTECore.rdf#"
    xmlns:aconf="http://artiste.it-
    innovation.soton.ac.uk/rdf/ARTISTEConfiguration.rdf#"
    xmlns:z19="http://artiste.it-
    innovation.soton.ac.uk/rdf/thesaurus.rdf#">
  - <z19:Category rdf:ID="rectangle">
      <rdfs:label xml:lang="en">en:rectangle</rdfs:label>
      <rdfs:label xml:lang="fr">fr:rectangle</rdfs:label>
    </z19:Category>
  - <z19:Category rdf:ID="triptych">
      <rdfs:label xml:lang="en">en:triptych</rdfs:label>
      <rdfs:label xml:lang="fr">fr:triptych</rdfs:label>
    </z19:Category>
  - <z19:Category rdf:ID="square">
      <rdfs:label xml:lang="en">en:square</rdfs:label>
      <rdfs:label xml:lang="fr">fr:carre</rdfs:label>
    </z19:Category>
  :
  :
  </rdf:RDF>
```

Figure 3-4Single Attribute Thesaurus for the dynamically generated metadata element Bordershape


ARTISTE - Art Analysis and Navigation Environment - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://ngl.artiste.it-innovation.soton.ac.uk:8080/servlet/CommandControllerServlet?command=setLocale&lang=en&country=GB
New Search
Search History
?

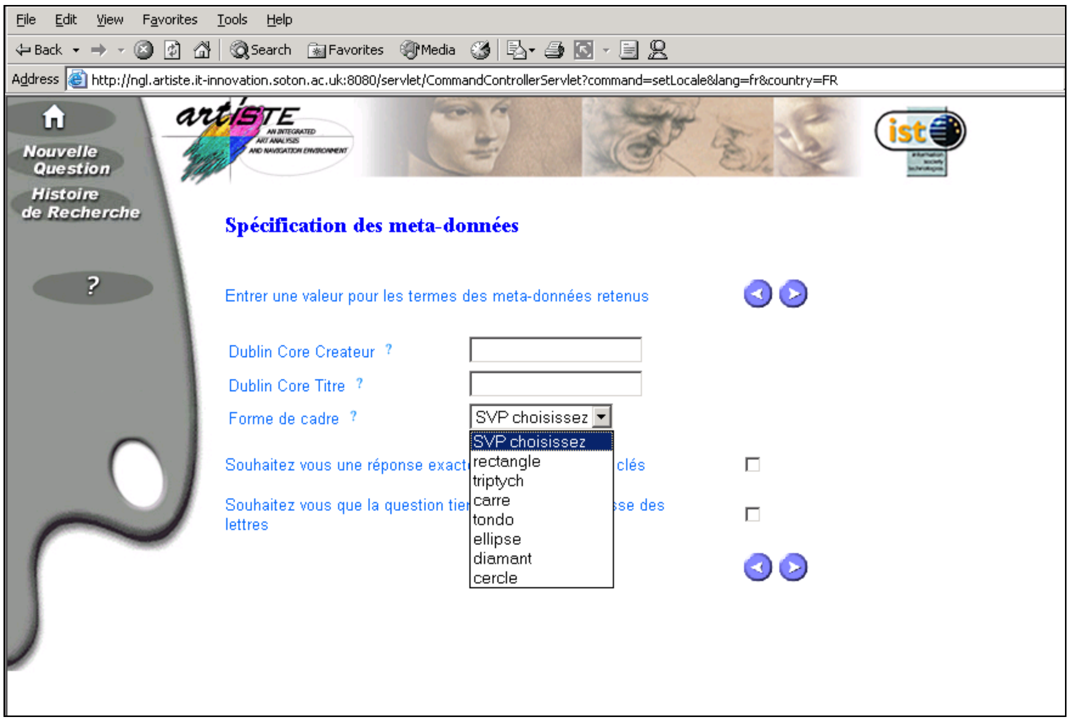
Metadata Specification

Enter values for the metadata terms of interest:

Border Shape ?	Please Select
Dublin Core Creator ?	
Dublin Core Title ?	
Would you like an exact match for	☐
Would you like the search to be ca	☐

Please Select
rectangle
triptych
square
tondo
ellipse
diamond
circle

Figure 3-5English Language using thesaurus for dynamically generated metadata element.


File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://ngl.artiste.it-innovation.soton.ac.uk:8080/servlet/CommandControllerServlet?command=setLocale&lang=fr&country=FR
Nouvelle Question
Histoire de Recherche
?

Spécification des meta-données

Entrer une valeur pour les termes des meta-données retenus

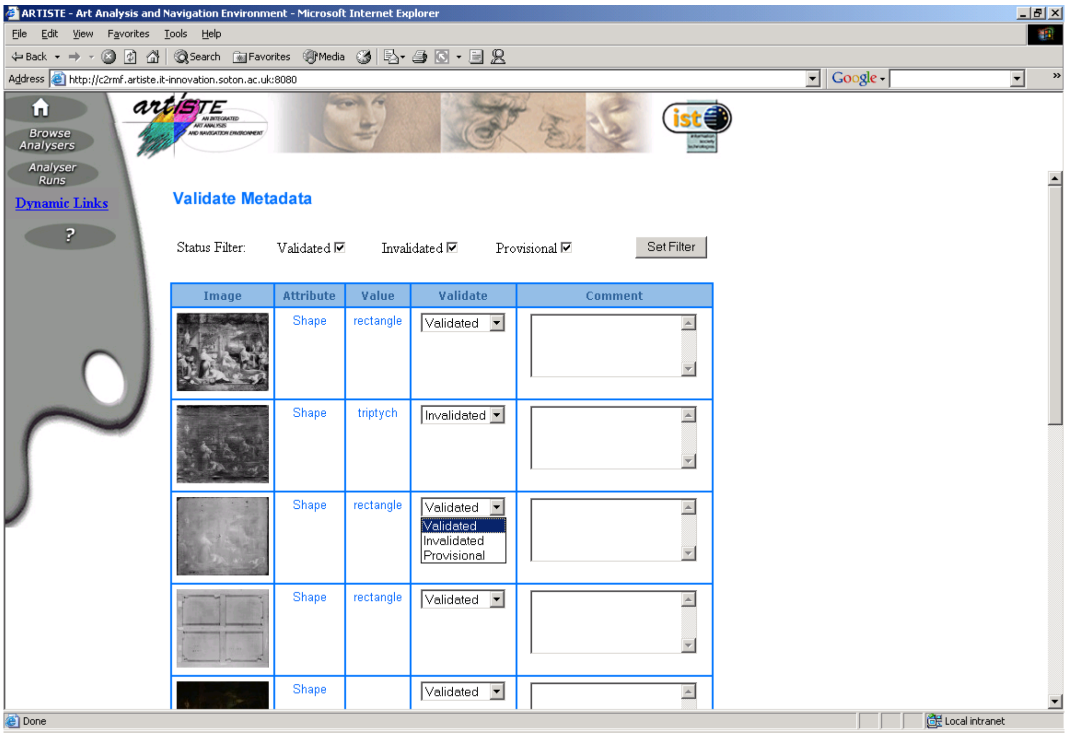
Dublin Core Createur ?	
Dublin Core Titre ?	
Forme de cadre ?	SVP choisissez
Souhaitez vous une réponse exact clés	☐
Souhaitez vous que la question tie se des lettres	☐

SVP choisissez
rectangle
triptych
carre
tondo
ellipse
diamant
cercle

Figure 3-6French Language interface using thesaurus for dynamically generated metadata element.

3.3 Metadata Validation

ARITSTE provides tools to validate dynamically generated metadata as shown in Figure 3-7.


ARTISTE - Art Analysis and Navigation Environment - Microsoft Internet Explorer

Address http://c2rmf.artiste.it-innovation.soton.ac.uk:8080

Validate Metadata

Status Filter: Validated ☑ Invalidated ☑ Provisional ☑ Set Filter

Image	Attribute	Value	Validate	Comment
	Shape	rectangle	Validated	
	Shape	triptych	Invalidated	
	Shape	rectangle	Validated (Validated / Invalidated / Provisional)	
	Shape	rectangle	Validated	
	Shape		Validated	

File Edit View Favorites Tools Help
Back Search Favorites Media
Browse Analysers
Analyser Runs
Dynamic Links
?
Done
Local intranet

Figure 3-7ARTISTE metadata validation tool.

4. Open Archive Initiative

4.1 Open Archive Initiative Metadata Harvesting Protocol

The goal of the OAI harvesting protocol is to supply and promote an application-independent interoperability framework that can be used by a variety of communities engaged in publishing content on the Web. ARTISTE is an OAI data provider and has implemented support for the Open Archives Initiative Protocol for Metadata Harvesting, thus providing open access to metadata stored with each museum and gallery collection.

4.2 Implementation of Support for OAI-PMH

4.2.1 Repository

In OAI terminology each installation of the ARTISTE system acts as a metadata repository. This is defined as a network accessible server that can process the 6 OAI-PMH requests:

- Identify
- List Metadata Formats
- List Identifiers
- List Records
- List Sets
- Get Record

Thus each Site (Victoria & Albert Museum, C2RMF, National Gallery, Uffizi) is enabled to act as OAI Repository

The OAI-PMH distinguishes between three distinct entities related to the metadata made accessible by the OAI-PMH in a repository: resource, item and record.

4.2.2 Resources

A resource is the object or "stuff" that metadata is "about". The nature of a resource, whether it is physical or digital, or whether it is stored in the repository or is a constituent of another database, is outside the scope of the OAI-PMH. In ARTISTE the resources are the images stored in the database. These are never returned to the client as the result of an OAI-PMH request.

4.2.3 Item

An item is a constituent of a repository from which metadata about a resource can be disseminated. Conceptually an item is a container that stores or dynamically generates metadata about a single resource in multiple formats, each of which can be harvested as records via the OAI-PMH. In the ARTISTE implementation of the OAI-PMH, items are conceptually equivalent to the unique identifiers stored in the databases for each of the images since all the metadata associated with an image (or resource) is accessed via this identifier.

Each item has an identifier that is unique within the scope of the ARTISTE OAI repository of which it is a constituent. This unambiguously identifies an item within the repository and is used in OAI-PMH requests for extracting metadata from the item. It would therefore be possible to use the existing TOR database image_id as the item identifier for OAI. However the OAI-PMH specifies that the format of a unique ID for records must correspond to that of the URI syntax. The OAI-PMH specifies further formatting (in an XML Schema named 'oai-identifier') which is recommended but not required. The advantage of adopting this naming convention is that identifiers are resolvable via a central OAI resolution service.

To comply with the 'oai-identifier' format the unique IDs must be comprised of

A "scheme" = oai

A "repositoryIdentifier" that is a unique ID for a repository.

A "local ID" that is the unique ID of a record within a repository

These three parts must be concatenated using a "delimiter" which must be a colon.

The ARTISTE implementation of support for OAI_PMH therefore includes the generation of oai-identifiers for all the images in the ARTISTE system. These take the form:

oai:artiste:SiteName/ImageID

For example:

oai:artiste:c2rmf/13640

4.2.4 Record.

A record is metadata in a specific metadata format. A record is returned as an XML-encoded byte stream in response to a protocol request to disseminate a specific metadata format from a constituent item. The XML-encoding of records is organized into the following parts:

- *header* -- contains the unique identifier of the item and properties necessary for selective harvesting. The header consists of the following parts:
  - the unique identifier -- the unique identifier of an item in a repository;
  - the date stamp -- the date of creation, modification or deletion of the record for the purpose of selective harvesting.
- *metadata* -- a single manifestation of the metadata from an item. The OAI-PMH supports items with multiple manifestations (formats) of metadata. The specific

metadata format of the record to be disseminated is specified by means of an argument -- the `metadataPrefix` -- in the `GetRecord` or `ListRecords` request that produces the record. The `ListMetadataFormats` request returns the list of all metadata formats available from a repository, or for a specific item (which can be specified as an argument to the `ListMetadataFormats` request).

Each of ARTISTE OAI Repositories support the dissemination of records in the OAI Dublin Core metadata format as well as the legacy formats described in the individual Collection schemas.

4.2.5 Sets

A OAI set is an optional construct for grouping items for the purpose of selective harvesting. The ARTISTE OAI repositories do not contain sets.

4.2.6 HTTP Embedding of OAI-PMH requests

OAI-PMH requests are expressed as HTTP requests. OAI-PMH requests to the ARTISTE server can be submitted using either the HTTP `GET` or `POST` methods. There is a single base URL for all requests. The ARTISTE OAI Repositories expose their base URL, as the value of the `baseURL` element in the `Identify` response.

The baseURL specifies the ARISTE host and port, and the path to the OAI Servlet.

In addition to the base URL, all requests consist of a list of *keyword arguments*, which take the form of `key=value` pairs. Arguments may appear in any order and multiple arguments must be separated by ampersands. Each OAI-PMH request must have at least one `key=value` pair that specifies the OAI-PMH request issued by the harvester:

`key` is the string '`verb`';

`value` is one of the defined OAI-PMH requests .

The number and nature of additional `key=value` pairs depends on the arguments for the individual request.

The OAIServlet is initialised and accepts the GET or POST request. The servlet creates a new instance of the ARTISTE OAI Harvester using JNDI and a JDBC/ODBC connection. The servlet also creates a new instance of a SAX2 filter (XML Writer [xv]) that serializes its events to an XML document. The ARTISTE OAI Harvester is then passed the XML Writer object to which it streams the response XML.

4.2.7 Response Format

Responses to requests are formatted as HTTP responses, with appropriate HTTP header fields. The `Content-Type` returned for all OAI-PMH requests is `text/xml`.

All responses to OAI-PMH requests are well-formed XML instance documents. Encoding of the XML uses the UTF-8 representation of Unicode. The responses validate against the XML Schema for Validating Responses to OAI-PMH Requests [xvi] .

The example response to a GetRecord request in Figure 4-1 shows an XML-encoding of a record and its components:

- the *header* part with:
    - a unique identifier of the item from which the record was disseminated, equal to oai:artiste:Sample/46518;
    - the datestamp of the record equal to 2002-02-28;
- the *metadata* part. This consists of a single root tag - in the example the tag `oai_dc`- with the nested tags belonging to the corresponding metadata format -- in the example, Dublin Core elements such as `title`. Note that the root tag within the metadata part includes a number of attributes that are common to all XML documents that use namespaces and schema validity:
    - *namespace declarations* -- the declarations of the namespaces used within the metadata part, each of which is prefixed with `xmlns` . Namespace declarations within the metadata part fall into two categories:
        - *metadata format specific namespace(s)* - every metadata part must include one or more `xmlns` prefixed attributes that define the correspondence between a metadata format prefix -- e.g. `dc` -- and the namespace URI (as defined by the XML namespace specification) of the respective metadata format. Some metadata formats employ tags from multiple namespaces, requiring multiple `xmlns` prefixed attributes -- in the example, there is only a single declarations for `oai_dc` .
        - *xml schema namespace* - every metadata part must include the attribute `xmlns:xsi`, the value of which must always be the URI shown in the example, which is the namespace URI for XML schema.
        - `xsi:schemaLocation` -- the value of which is a URI, URL pair; the first is the namespace URI (as defined by the XML namespace specification) of the metadata that follows in this part, and the second is the URL of the XML schema for validation of the metadata that follows.

```
<?xml version="1.0" encoding="UTF-8" ?>
-<GetRecord xmlns="http://www.openarchives.org/OAI/1.1/OAI_GetRecord"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.openarchives.org/OAI/1.1/OAI_GetReco
    rd http://www.openarchives.org/OAI/1.1/OAI_GetRecord.xsd">
    <responseDate>2002-07-31T12:37:17+01:00</responseDate>
    <requestURL>http://artiste.it-
      innovation.soton.ac.uk/servlet/OAIServlet?verb=GetRecord</reque
      stURL>
  <record>
    <header>
        <identifier>oai:artiste:Sample/46518</identifier>
        <datestamp>2002-07-31T12:37:26+01:00</datestamp>
      </header>
    <metadata>
      <oai_dc xmlns="http://purl.org/dc/elements/1.0/"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://artiste.it-
          innovation.soton.ac.uk/rdf/oai_dc.rdf
          http://artiste.it-
          innovation.soton.ac.uk/rdf/oai_dc.xsd">
          <contributor>taylorc</contributor>
          <identifier>pcd228810120146-026</identifier>
          <description>An Indian black buck is shown being led
            by it's keeper. The border of the image is a floral
            design and watercolour and gold on
            paper.</description>
          <subject>INDIAN PAINTING</subject>
          <source>CT1755</source>
          <title>Indian Black BuckMughalc.1615No Date</title>
          <creator>Indian Black BuckMughalc.1615No
            Date</creator>
        </oai_dc>
      </metadata>
    </record>
  </GetRecord>
```

Figure 4-1Example response to a GetRecord request to the OAI Repository

4.3 OAI Clients

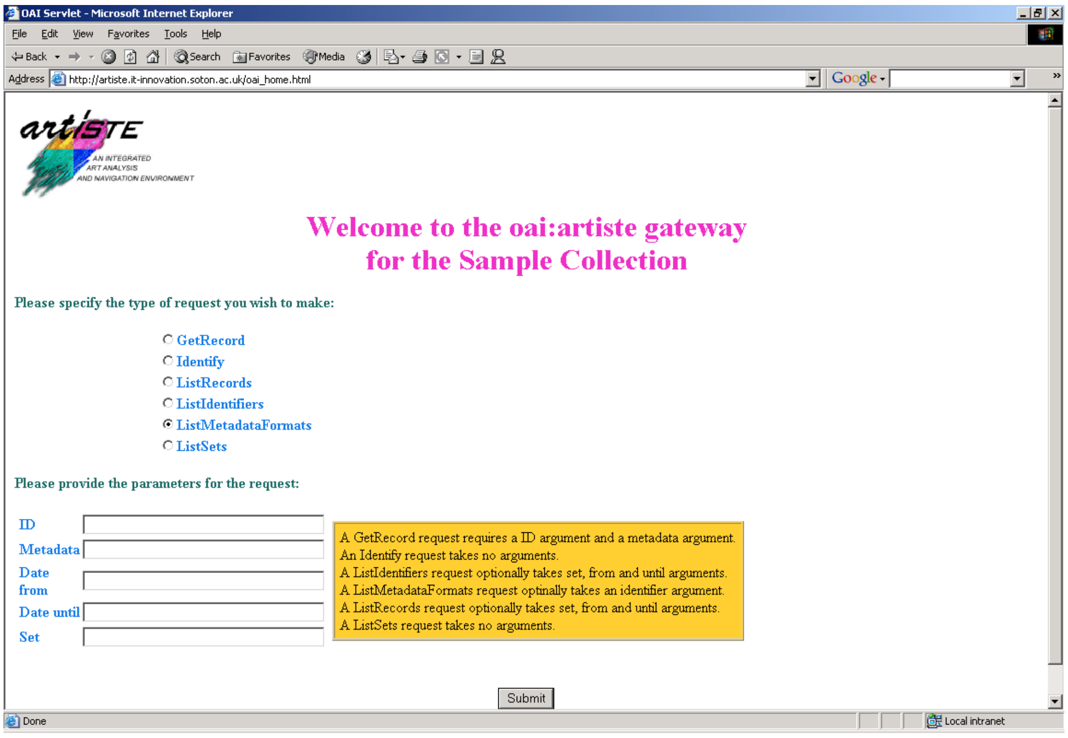
Each ARTISTE OAI repository can be access in three ways

1. Issuing an HTTP Get request directly via a web browser using the base URL http://artiste.it-innovation.soton.ac.uk/servlet/OAIServlet eg

   http://artiste.it-innovation.soton.ac.uk/servlet/OAIServlet?verb=GetRecord&identifier=oai:arXiv:hep-th/9901001&metadataPrefix=oai_dc

2. Via a dedicated ARTISTE OAI Web Interface. The OAI Web Interface to the ARTIST dissemination system is available at http://artiste.it-innovation.soton.ac.uk/oai_home.html .

Figure 4-2 shows an example ListMetadataFormats request being issues via the ARTISTE OAI Web Interface to the Sample Dissemination system. Figure 4-3 shows the resulting XML response.


OAI Servlet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://artiste.it-innovation.soton.ac.uk/oai_home.html
Done
Local intranet

Welcome to the oai:artiste gateway
for the Sample Collection

Please specify the type of request you wish to make:

- GetRecord
- Identify
- ListRecords
- ListIdentifiers
- [✓ ListMetadataFormats]
- ListSets

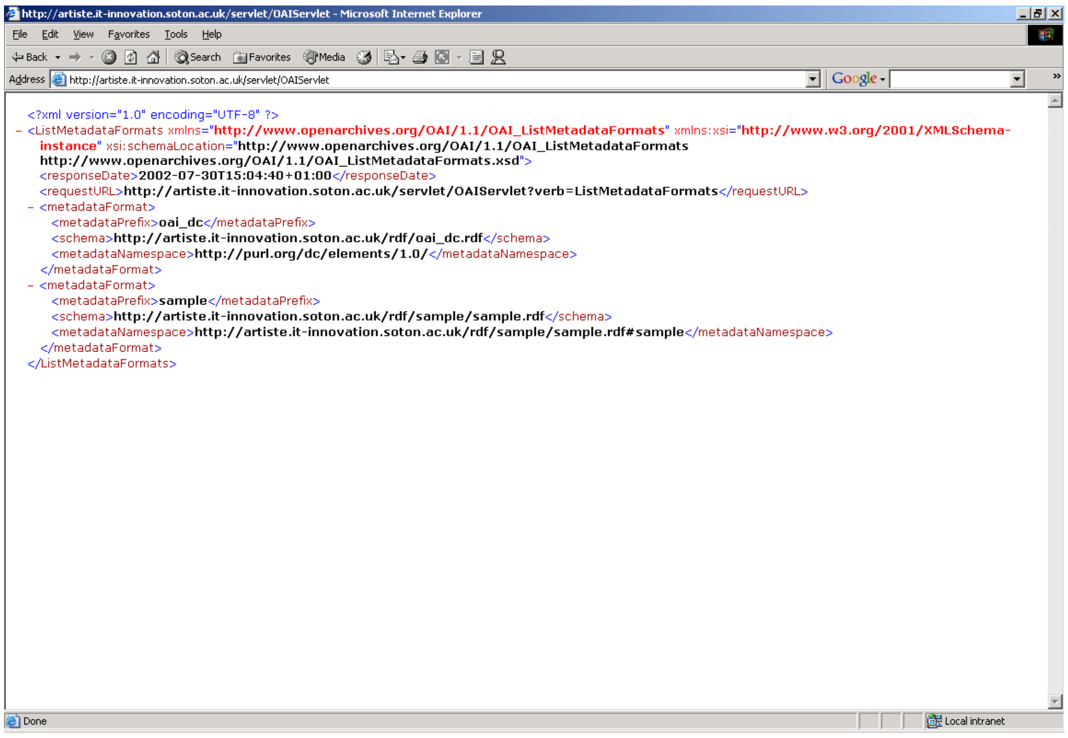
Please provide the parameters for the request:

ID
Metadata
Date from
Date until
Set

A GetRecord request requires a ID argument and a metadata argument.
An Identify request takes no arguments.
A ListIdentifiers request optionally takes set, from and until arguments.
A ListMetadataFormats request optinally takes an identifier argument.
A ListRecords request optionally takes set, from and until arguments.
A ListSets request takes no arguments.

Submit

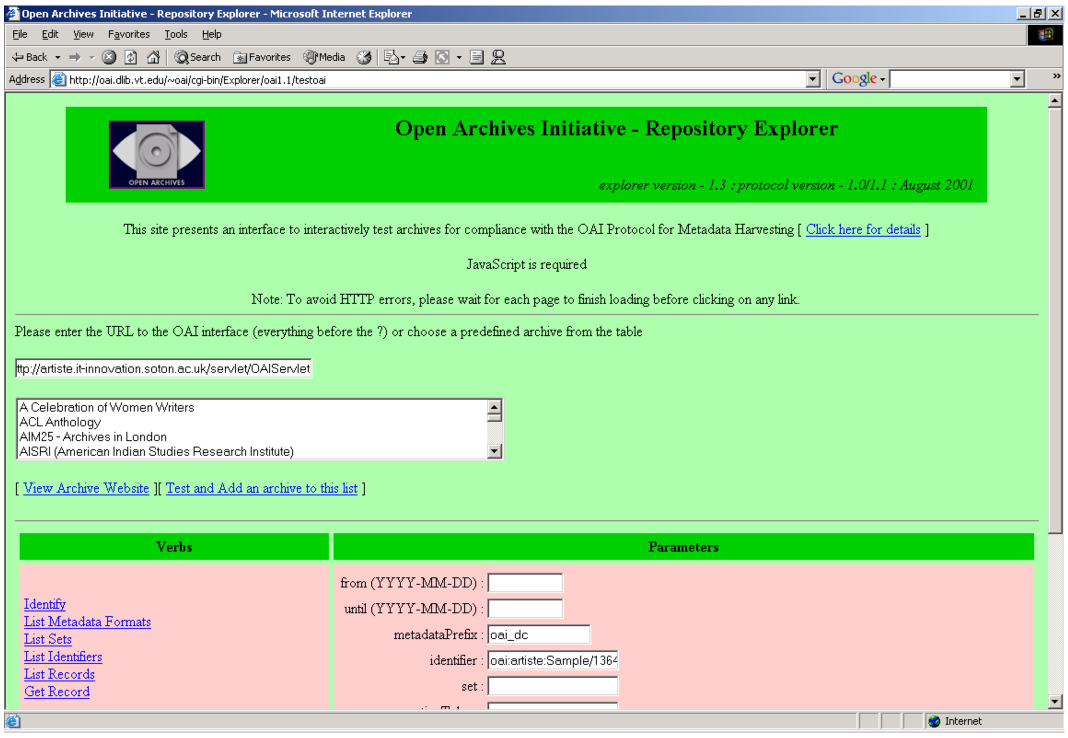
Figure 4-2A ListMetadataFormats request being issued via the web interface to the Sample ARTISTE OAI Repository.



```
<?xml version="1.0" encoding="UTF-8" ?>
- <ListMetadataFormats xmlns="http://www.openarchives.org/OAI/1.1/OAI_ListMetadataFormats" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
  instance" xsi:schemaLocation="http://www.openarchives.org/OAI/1.1/OAI_ListMetadataFormats
  http://www.openarchives.org/OAI/1.1/OAI_ListMetadataFormats.xsd">
  <responseDate>2002-07-30T15:04:40+01:00</responseDate>
  <requestURL>http://artiste.it-innovation.soton.ac.uk/servlet/OAIServlet?verb=ListMetadataFormats</requestURL>
- <metadataFormat>
    <metadataPrefix>oai_dc</metadataPrefix>
    <schema>http://artiste.it-innovation.soton.ac.uk/rdf/oai_dc.rdf</schema>
    <metadataNamespace>http://purl.org/dc/elements/1.0/</metadataNamespace>
  </metadataFormat>
- <metadataFormat>
    <metadataPrefix>sample</metadataPrefix>
    <schema>http://artiste.it-innovation.soton.ac.uk/rdf/sample/sample.rdf</schema>
    <metadataNamespace>http://artiste.it-innovation.soton.ac.uk/rdf/sample/sample.rdf#sample</metadataNamespace>
  </metadataFormat>
</ListMetadataFormats>
```

Figure 4-3 The XML response to a ListMetadataFormats request to the Sample ARTISTE OAI Repository.

3. Via the official OAI Repository Explorer online at http://oai.dlib.vt.edu/~oai/cgi-bin/Explorer/oai1.1/testoai as shown in Figure 4-4 .


Open Archives Initiative - Repository Explorer - Microsoft Internet Explorer

Address http://oai.dlib.vt.edu/~oai/cgi-bin/Explorer/oai1.1/testoai

Open Archives Initiative - Repository Explorer

explorer version - 1.3 : protocol version - 1.0/1.1 : August 2001

This site presents an interface to interactively test archives for compliance with the OAI Protocol for Metadata Harvesting [Click here for details]

JavaScript is required

Note: To avoid HTTP errors, please wait for each page to finish loading before clicking on any link.

Please enter the URL to the OAI interface (everything before the ?) or choose a predefined archive from the table

ttp://artiste.it-innovation.soton.ac.uk/servlet/OAIServlet

A Celebration of Women Writers
ACL Anthology
AIM25 - Archives in London
AISRI (American Indian Studies Research Institute)

[View Archive Website][Test and Add an archive to this list]

Verbs	Parameters
Identify List Metadata Formats List Sets List Identifiers List Records Get Record	from (YYYY-MM-DD) : until (YYYY-MM-DD) : metadataPrefix : oai_dc identifier : oai:artiste:Sample/1364 set :

Figure 4-4 A GetRecord request being issued to the Sample ARTISTE OAI Repository via the official OAI Repository Explorer

5. Distributed Query Layer

ARTISTE is participating in an initiative to redesign the primary open standard for interoperability between digital libraries, z39.50 [xvii], using web technologies such as XML and SOAP. The z39.50 into the Next Generation (ZING) initiative has proposed a Search and Retrieve Web Service (SRW) based on the z39.50 protocol for searching databases that contain metadata and objects.

ARTISTE is one of the early implementers of SRW and has extended the capabilities of SRW to enable image content and metadata based searches over multiple ARTISTE collections. Having emerged from the digital library community z39.50 has been traditionally concerned with text based searching. ARTISTE has been working with ZING to incorporate the ability to deal with content-based searching of images and thus expand international standards of information retrieval.

The implemented support for an extended version of the SRW protocol forms the basis of a 'distributed query layer'(DQL) through which the ARTISTE servers communicate with each other as illustrated in Figure 1-2.

5.1 Z39.50

The z39.50 protocol specifies formats and procedures governing the exchange of messages between a client and server enabling the client to request that the server search a database and identify records which meet specified criteria, and to retrieve some or all of the identified records. In other words it provides a framework for distributed queries.

A Z39.50 server must support a core of functions: initialisation, search and retrieval. The basic process of querying is shown in Figure 5-1.

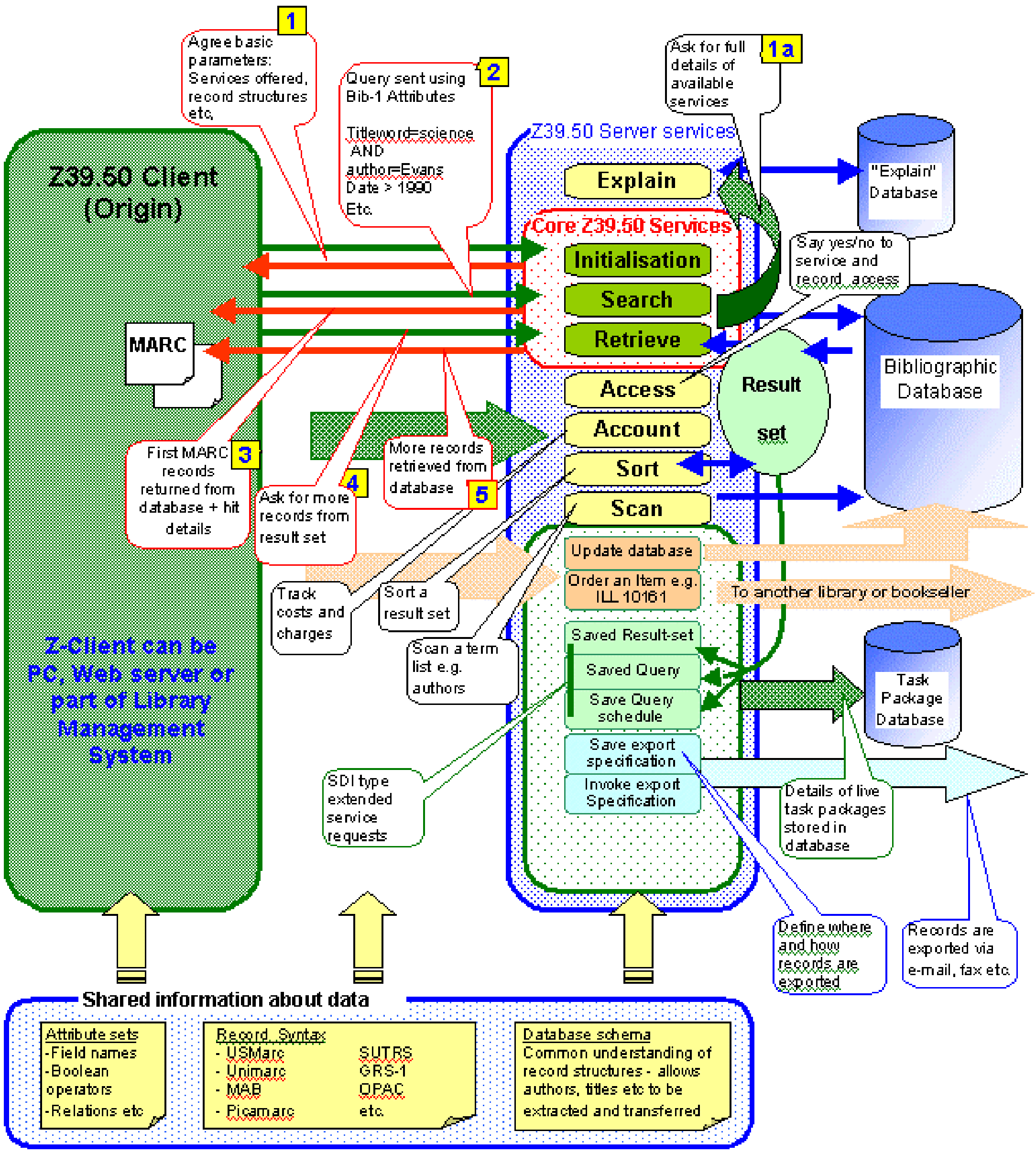

1
Agree basic parameters: Services offered, record structures etc.
2
Query sent using Bib-1 Attributes
Titleword=science AND author=Evans Date > 1990 Etc.
Ask for full details of available services
1a
Z39.50 Client (Origin)
Z39.50 Server services
Explain
"Explain" Database
Core Z39.50 Services
Initialisation
Search
Retrieve
Say yes/no to service and record access
MARC
Bibliographic Database
Access
Result set
Account
Sort
Scan
First MARC records returned from database + hit details
3
Ask for more records from result set
4
More records retrieved from database
5
Update database
Order an Item e.g. ILL 10161
To another library or bookseller
Track costs and charges
Sort a result set
Scan a term list e.g. authors
Saved Result-set
Saved Query
Save Query schedule
Task Package Database
Z-Client can be PC, Web server or part of Library Management System
Save export specification
Invoke export Specification
SDI type extended service requests
Details of live task packages stored in database
Define where and how records are exported
Records are exported via e-mail, fax etc.
Shared information about data
Attribute sets -Field names -Boolean operators -Relations etc
Record Syntax - USMarc - Unimarc - MAB - Picamarc SUTRS GRS-1 OPAC etc.
Database schema Common understanding of record structures - allows authors, titles etc to be extracted and transferred

Figure 5-1 Overview of z39.50 functionality. Diagram from Biblio Tech Review (http://www.biblio-tech.com/html/z39_50.html)

The underlying protocol generally adopted for Z39.50 is TCP/IP.

As an investigation of the usefulness of the z39.50 protocol to the ARTISTE project a test z39.50 client and server were built using the Open Source Java toolkit JZKit from Knowledge Integration [xviii] It was not expected that the z39.50 server would directly execute a search on the database. Instead the z39.50 server would issue a request to the ARTISTE server which would perform the search and return the appropriate result to the z39.50 server. It would then be up to the z39.50 server to return appropriate results to the z39.50 client

The functionality provided by the ARTISTE system can be described using 6 broad categories. These are:

i) Metadata Querying

ii) Image Querying

iii) Complex Querying

iv) Saving and Manipulating Queries

v) Cataloguer Functions

vi) Administrator Functions

Categories v and vi are not query based and the functionality they describe would not be supported by a distributed query layer. Therefore the evaluation of z39.50 focused on support for categories i-iv. The conclusions drawn from the evaluation are summarized in Table 1

Table 1 Evaluation of support for ARTISTE DQL functionality by z39.50

ARTISTE metadata and image content based searching functionality	**Supported by z39.50 protocol**
Metadata Query	Y
Image Query	N
Query Across Multiple Databases	Y
Query Across Multiple Sites	N
Query Using Multiple Metadata Terms	Y
Query Using Metadata Terms & A Query Image	N
Query Using Controlled Metadata Terms	N
Query Using Multilingual Metadata Terms	N
Save a Query	N
Delete a Query	N
Browse Query Runs and Retrieve Results for a Query	N
Support dynamics of new & changing metadata	N

5.2 ZING SRW

ZING (Z39.50 International: Next Generation) is an umbrella term for a series of initiatives that are being pursued by the z39.50 community. The ZING aims to make the semantic content of Z39.50 more broadly available and to make Z39.50 more attractive to information providers, developers, vendors, and users, by lowering the barriers to implementation while preserving the existing intellectual contributions of Z39.50.

SRW is the ZING Search/Retrieve Web Service which defines a draft framework for query and retrieval that takes the core search and retrieval protocol from z39.50 and specifies a Web Service implementation.

The SRW protocol retains many concepts from the traditional z39.50 protocol including Result Sets, Abstract Access points, Abstract Record schemas, Explain and Diagnostics. However it differs significantly from z39.50 in several ways which make it a better candidate for the ARTISTE DQL that 'pure' or 'traditional' z39.50.

5.2.1 SRW Features which Differ from Z39.50

- **Result Set Named by Server**

In contrast to Z39.50 where the client names the result set, for SRW the server assigns the result set id. After a server executes a query it may include in the response a result set name. This coincides with the ARTISTE concept of a result set which is named by a query_run_id. This result set persists in the underlying database and can be accessed multiple times.

- **Connections, Sessions, State**

There is no explicit concept of connection, session, or state. Each invocation of the Search/Retrieve service will be a request/response sequence, via an XML/SOAP/RPC message using HTTP POST. The most notable of the differences between SRW and z39.50 is that queries are specified using the Common Query Language, and XML messages are exchanged between client and server using SOAP. The use of XML and SOAP makes it much simpler to develop SRW clients and servers than Z-Client and Z-Servers since tools that support these technologies are readily available off the shelf.

- **No distinction between server and database**

SRW does not distinguish between a server and a database. A single SRW request can therefore be sent to multiple sites and/or multiple databases.

- **Single record syntax**

All SRW records are retrieved according to a single record syntax (XML) and therefore the Z39.50 concept of record syntax is not meaningful in SRW. The Z39.50 concepts of element set/specification and schema are represented by XML schemas. The following record *schemas* are distinguished in SRW: Dublin Core, Onix, MODS, and MarcXml. The use of XML as a record syntax means that the SRW responses can be encoded in RDF/RDFS and thus interoperate with the ARTISTE query ontology defined in ARTISTE Core as well as the various Collection schemas.

- **String Query**

SRW specifies string queries. The query language, CQL ("Common Query Language"), is a human-readable-string query-representation based loosely on CCL (however just the query, no commands) with access points defined.

- **Flat Access Points**

Flat access points are defined, rather than utilizing attribute vectors as in traditional Z39.50. Again this means a better match with the ARTISTE RDF access points which do not use attribute vectors.

- **Static Explain**

Explain information will be static. The ARTISTE system was not designed with the provision of a z39.50 Explain function but the provision of a static set of data about the system is implementable within the existing architecture.

- **XML instead of ASN.1.**

XML is used for abstract syntax as well as encoding. ASN.1/BER is not used. Again the use of XML maps well into the ARTISTE architecture.

5.2.2 Limitations of SRW

However because the SRW initiative is primarily concerned with lowering the barriers to implementation of z39.50 it does not address all of deficiencies in z39.50 as a basis for an ARTISTE DQL as outlined above.

To support searching over multiple sites for example, it would be necessary to introduce an ARTISTE Distributed Query Layer Protocol (DQLP) that carries requests to an intermediate server that manages the breakdown of the DQLP query into multiple z39.50 SRW requests. The DQLP would most likely use SOAP as a transport protocol. The intermediate server would also be responsible for the collation of results into a results set, and the storage of queries and result sets.

It also seems unlikely that it will be possible to include the multi-lingual or controlled metadata terms in a DQL based on z39.50 SRW firstly because their use depends on control of the client, not something that is part of the ARTISTE DQL, and secondly because the SRW specification does not include services to query the server about which query items are supported. This functionality is available in the complete z39.50 specification and is know as ‘Explain ‘ but is very sparsely supported. The ‘Explain’ functionality in SRW is based on static information only and could not therefore interrogate the ARTISTE Query Context to discover lists of controlled values or multilingual labels for metadata attributes. This information is available to users via the published RDF Collection schemas.

Furthermore having emerged from the digital library community, z39.50 has been traditionally concerned with text based searching. As a result the CQL query language proposed by the SRW specification supports metadata based querying but makes no provision for content based queries. For example there is no provision to specify image analysers in combination with image operators and the search term is always assumed to be a text string. Conversely the CQL specification also allows for the formulation of more complex text queries than ARTISTE is capable of processing. For example the specifying left and/right truncation of terms is not possible within ARTISTE nor is the specification of proximity expressed in terms of "word", "sentence", "paragraph", or "element". Such functionality is not available in the TOR database and therefore not in the ARTISTE application server.

Finally the SRW protocol is limited because of its draft status. A pre-release of version 1.0 is expected to be available in mid-August 2002 and release 1.0 is due be announced in early October but at the time of writing the specifications are incomplete and under frequent revision.

These limitations were addressed by extending the protocol in those areas where it does not cover image content based querying (notably CQL), and by maintaining close contact with the members of the ZING group developing the SRW specifications, both via the ZING mailing list and at various face to face meetings. As an early implementer of the draft SRW specification [xix]. ARTISTE has given feedback to the community on implementation and modification of SRW.

5.3 ARTISTE SRW Server

The ARTISTE SRW runs on the Tomcat JSP/servlet container engine[xx] under Apache [xxi]. The ARTISTE SRW Server receives an request from a client in the form of an XML message via SOAP/RPC. A W3C submission has proposed an extension to the SOAP bindings to enable a SOAP 1.1 message to be carried within a MIME multipart/related message [xxii]. However this is not well supported by existing SOAP toolkits has not been adopted as part of the SRW protocol so it is not possible to actually embed a query image in the message sent to the SRW Server. Images are therefore passed to and from the SRW server by reference using URIs. The SRW Service parses the request parameters and uses the query ontology described in the ARTISTE Core RDF schema (Section 2.2.1) to build a corresponding ARTISTE query. If the SRW request contains an image content based query the ARTISTE SRW Server retrieves the query image from the specified web server and passes it, along with the RDF specified query, to the ARTISTE application server. The ARTISTE application server translates the query to the appropriate SQL and executes it against the database. The results are returned to the SRW Server which then makes another call on the ARTISTE application server to extract the metadata associated with the returned results in either the default metadata format (Dublin Core) or another format specified in the SRW request. The SRW Server the responds to the SRW request by sending an XML message containing the URL of image results and RDF encoded metadata back to the client.

The rest of this section describes in more detail the ARTISTE implementation of the SRW protocol.

5.3.1 Request Parameters

5.3.1.1 Query

Optional

If neither a query nor a result set id argument is received by the Server the request is interpreted as an 'Explain' request.

If both a query and a result set id argument are received by the Server an error message is returned. – i.e. it is not possible to execute a query over a subset of the collection.

May be supplied on its own or accompanied any of the following combinations

- record schema, start record, maximum records
- record schema, start record
    - the Server decides how may records to return. The ARTISTE SRW returns 50 records by default
- record schema, maximum records
    - the start record is assumed to be 1
- record schema
    - the Server decides how may records to return. The ARTISTE SRW returns 50 records by default.
    - the start record is assumed to be 1
- start record, maximum records
    - the Server returns records using a default schema. The ARTISTE SRW default schema is Dublin Core
- start record
    - the Server returns records using a default schema. The ARTISTE SRW default schema is Dublin Core. The Server decides how may records to return.
    - The ARTISTE SRW returns 50 records by default
- maximum records
    - the Server returns records using a default schema. The ARTISTE SRW default schema is Dublin Core.
    - the start record is assumed to be 1

The implemented ARTISTE SRW accepts queries specified in either the standard CQL syntax or the extended version of the syntax known as CAQL (Common ARTISTE Query Language)

If an invalid query string is received an error message is returned

If a valid query string CQL is received which employs those elements of the CQL syntax not supported by ARTISTE an error message is returned.

The draft SRW specification states that

The server, upon receiving the request, might execute the query, or might decide that there is already a stored result set corresponding to the supplied query string ... The server may decide to use retained results rather than re-execute the query, and this decision is entirely at the server's discretion and is transparent to the protocol. (http://www.loc.gov/z3950/agency/zing/srw.html)

Upon receipt of a valid query request the ARTISTE SRW always executes that query. If the client wants to retrieve results from a stored result set they should supply a resultSetId not a query.

5.3.1.2 Result Set Id

Optional

Again if neither a query nor a result set id argument is received by the Server the request is interpreted as an 'Explain' request.

Again if both a query and a result set id argument are received by the Server an error message is returned. – i.e. it is not possible to execute a query over a subset of the collection.

May be supplied on its own or accompanied any of the combinations listed above for Query

The Result Set Id must have been supplied by the server in an earlier response.

The ARTISTE SRW uses query_run_ids as result set ids.

If an invalid result set id is received (i.e. one for which there is no corresponding record in the database) an error message is returned. If a valid result set id is received which contains no results it is processed as normal.

5.3.1.3 Record Schema

Optional.

Ideally the reference to the record schema would be a URI

e.g. http://artiste.it-innovation.soton.ac.uk/rdf/vam/vam.rdf

However the SRW specification mandates single word identifiers such as DC, ONIX or, MODS . The ARTISTE SRW Servers support reference to record schemas either by identifier or by URI.

All ARTISTE SRW Servers support the oai_dc record schema. Individual SRW Server also support the record schemas based on the Collection thesauri for example vam (URI reference http://artiste.it-innovation.soton.ac.uk/rdf/vam/vam.rdf) or ngl (URI reference http://artiste.it-innovation.soton.ac.uk/rdf/ngl/ngl.rdf.) The information about which records schemas a Server supports is read in from the RDF Collection schemas defining the underlying collection.

If a reference to a schema not supported by the ARTISTE SRW is received an error message is returned.

5.3.1.4 Start Record

Optional

If omitted Start Record is set to 1.

5.3.1.5 Maximum Records

Optional

If this is set to zero then no records are returned.

If omitted Maximum Records is set to 50.

The ARTISTE SRW imposes a limit on the total number of records that can be returned an a single response. This limit is set to 100. If the server receives a request for more than 100 records an error message is returned. This constraint is not part of the draft SRW protocol but following an ARTISTE initiated discussion on the ZING mailing list there was general approval for its inclusion.

5.3.1.6 Session Id

Optional

The ARTISTE SRW does not support session ids. If the Server receives a session id an error message is returned.

5.3.2 Response Parameters

5.3.2.1 Response to a valid SRW

The response to a valid SRW request contains:

- Result Set
- Result Set Id
- Number of Records
- Records

The draft SRW specification states that the Result Set also contains

Result Set Idle Time

The ARTISTE SRW does not support this and it is not returned as part of a response.

The draft SRW specification states that

In each Search/Retrieve response the server may include a result set idle time value indicating a projected (not guaranteed) length of time that the result set will remain available if it is not referenced. Once the result set is referenced in a subsequent Search/Retrieve request, that response may include a new value for the result set idle time. (http://www.loc.gov/z3950/agency/zing/srw.html)

The ARTISTE SRW does not include a result set idle time in the response

5.3.2.2 Response to an Explain Request

An explain request returns

- Explain
- Comment
- Record Schemas

An invalid request returns

5.3.3 Diagnostics

The draft SRW specification states that

The Z39.50 tradition of providing rich diagnostics is retained in SRW. Thus application level diagnostics will be defined, not just mappings to lower-level fault codes. (http://www.loc.gov/z3950/agency/zing/srw.html)

The intention is to use the Bib-1 Diagnostics defined within the Z39.50-1995 Standard (http://lcweb.loc.gov/z3950/agency/defns/bib1diag.html)

The ARTISTE SRW has not fully implemented the Bib-1 codes but does return diagnostic error messages according to the suggested schema in http://www.loc.gov/z3950/agency/zing/service.html

```
<diagnostic>
<code> code </code>
<text> text </text>
<addInfo> additional information </addInfo>
</diagnostic>
```

The Bib-1 diagnostics supported by the ARTISTE SRW are listed in Table 2. In other cases errors are reported as code 999 and an ARTISTE SRW determined Diagnostic Meaning. These diagnostics are listed in Table 3

Table 2Bib-1 Diagnostics

Code	Meaning	Additional Info
30	Specified result set does not exist	(unspecified)
108	Malformed Query	(unspecified)

Table 3ARTISTE Diagnostics

Code	Meaning	Additional Info
999	ARTISTE SRW Diagnostic	The ARTISTE SRW does not support Session Ids

999	ARTISTE SRW Diagnostic	The ARTISTE SRW does not support Sort Keys.
999	ARTISTE SRW Diagnostic	A query request cannot be accompanied by a result set ID
999	ARTISTE SRW Diagnostic	Number or records requested exceeds the ARTISTE SRW limit. The ARTISTE SRW Server returns a maximum of DEFAULT_MAXIMUM_RECORDS records.
999	ARTISTE SRW Diagnostic	The ARTISTE SRW Server does not support the RECORD_SCHEMA_NAME record schema. A list of supported record schemas can be obtained by issuing an 'Explain' request

The SRW specification suggests (but does not mandate) non-fatal diagnostics. All errors are fatal in the ARTISTE SRW.

5.3.4 Common ARTISTE Query Language

This section provides a detailed breakdown of the SRW specified language CQL and describes those elements supported by the ARTISTE SRW and the extensions made to the syntax to enable image content based searching. The extended version of CQL is known as CAQL.

The CQL syntax begins :

```
cql-query ::= and-expr *( "or" and-expr).

and-expr ::= not-expr *( "and" not-expr ).

not-expr ::= base-expr *( "not" base-expr ).

base-expr ::= primary | "("cql-query ")".
```

The ARTISTE DQL cannot support any of these statements since ARTISTE does not support ‘OR’ or ‘NOT’ statements. Therefore the ARTISTE CAQL will support the following top level query syntax:

```
caql-query ::=base-expr *(“and” base-expr)

base-expr::=primary | caql-query
```

The SRW CQL syntax proceeds to specify

```
primary ::= result-set-expression | [index-name rel-op ] adj-expr
```

The ARTISTE CAQL expands this to provide support for image content queries

```
primary ::=result-set-expression | [index-name rel-op] adj-expr |
index-name img-op img-analyser img-exp
```

The SRW CQL specification of result-set-expression and index-name remains unchanged in the ARTISTE CAQL

The SRW CQL syntax specifies

```
rel-op ::= "=" | "<" | "<=" | ">" | ">=" | "<>"|"fuzzy"|
"stem"|"relevance"
```

The ARTISTE DQL only supports equals and contains operators, therefore the CAQL specifies

```
rel-op ::=″=″ | "<" | "<=" | ">" | ">="
```

The ARTISTE CAQL further specifies those elements necessary to an image content query

```
img-op ::= 〝SimilarTo″ | 〝PartOf″

img-analyser ::= identifier

img-expr ::= url
```

The SRW CQL includes the ability to specify "sameParagraph" and "sameSentence" queries. The ARTISTE does not support such queries. Nor does the ARTISTE support the 'OR' and 'NOT' statements used in the CQL `adj-primary`, `and-term`, and `not-term` elements. Therefore the ARTISTE CAQL specifies a much simpler version of the CQL `adj-expr`:

```
adj-expr ::=term *(〝and″ adj-expr)

term ::=identifier|quoted-string-literal
```

There is no similar concept in relation to image expressions because ARTISTE does not support the use of multiple image analysers in a single querying it is not possible to say "Find me image similar to the colour content of this image which are also similar to the colour content of this other image"

The entire BNF for the CAQL syntax is available in Appendix C.

5.3.4.1 Example CAQL Queries

dc.Title = advise "and" consent and bath.AuthorWord = drury (taken from SRW Example)

dc.Creator contains Vinci and artisteCore.VisibleLightImage SimilarTo CCV http://artiste.it-innovation.soton.ac.uk/test_images/test.jpg dc.Subject = TEXTILE and artisteCore.VisibleLightImage part of MCCV http://artiste.it-innovation.soton.ac.uk/test_images/test.jpg and dc.Creator contains Morris and William

The attribute/value part of the may be either an RDF URI pointing to an element as defined in a schema supported by ARTISTE e.g.

http://artiste.it-innovation.soton.ac.uk/test_rdf/dc.rdf#title

http://artiste.it-innovation.soton.ac.uk/test_rdf/vam/vam.rdf#PictureReference

Alternatively it may be a z39.50 IndexSet.IndexCode construction using the Bib-1 Attribute Set eg

Bib-1.1100

Bib-1.1103

In this case the Bib-1 code is mapped via a Bib-1 RDF schema (http://artiste.it-innovation.soton.ac.uk/test_rdf/bib1.rdf) to the local database schema.

5.4 Summary of SRW support for ARTISTE functionality

As illustrated in Table 4 although the SRW does not support all the elements of the ARTISTE metadata and image content based searching functionality it does expose a powerful subset of that functionality.

Table 4 Comparison of Artiste distributed search and retrieval functionality with ARTISTE SRW

ARTISTE metadata and image content based searching functionality	Supported by ARTISTE SRW
Metadata Query	Y
Image Query	Y
Query Across Multiple Databases	Y
Query Across Multiple Sites	N
Query Using Multiple Metadata Terms	Y
Query Using Metadata Terms & A Query Image	Y
Query Using Controlled Metadata Terms	N
Query Using Multilingual Metadata Terms	N
Save a Query	N
Delete a Query	N
Browse Query Runs and Retrieve Results for a Query	N

Support dynamics of new & changing metadata	Y

Furthermore some of those aspects of ARTISTE metadata and image content based searching functionality which are not directly supported by the SRW protocol are in fact indirectly enabled.

Firstly although the user cannot explicitly save or delete queries or browse query runs, the fact that the SRW Service returns a result_set_id as part of each response does mean that results for a query can be retrieved and the records iterated over. Secondly although each SRW Server communicates with a single site the protocol is designed to allow amalgamation of request to different SRW Servers. Thus a gateway to all the ARTISTE SRW Servers could be constructed (in fact this is the function performed by the ARTISTE application server which issues requests to the distributed databases via SRW and then collects the results before returning then to the user). Thirdly although the SRW interface does not explicitly expose the multilingual thesauri these are available on the web and users of the SRW are free to consult the RDF document in the creation of their queries.

The implementation of the SRW protocol therefore enables the ARTISTE system to execute metadata and image content based queries over distributed collections using a standard messaging and query protocol.

Equally importantly, support for the SRW protocol means that ARTISTE exposes a standard interface for the open query and retrieval of images across multiple collections.

5.5 ARTISTE SRW Client

The service can be accessed via standalone client which takes the following arguments via a command line interface.

-p	port number
-q	query
-k	record schema
-r	result set id
-b	start record
-m	maximum records
-j	session id
-c	sortKey

Figure 5-2and Figure 5-3 show a sample request and response to the ARTISTE SRW Server using the DQLCLient. Figure 5-4 shows one of the images returned in the response, accessed by following the URL returned in the <VisibleLightImage> element.

A public version of this client accessing the ARTISTE dissemination system is available at http://artiste.it-innovation.soton.ac.uk/servlet/DQLServlet.

```
POST /axis/servlet/AxisServlet HTTP/1.0

Content-Length: 1005

Host: localhost

Content-Type: text/xml; charset=utf-8

SOAPAction: ""

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

<SOAP-ENV:Envelope SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/">

 <SOAP-ENV:Body>

  <ns1:searchRetrieve xmlns:ns1="SrwService">

   <query xsi:type="xsd:string">http://artiste.it-
innovation.soton.ac.uk/test_rdf/oai_dc.rdf#subject = &apos;WEDDING DRESS&apos;
AND http://artiste.it-
innovation.soton.ac.uk/test_rdf/ARTISTECore.rdf#VisibleLightImage SimilarTo CCV
http://artiste.it-innovation.soton.ac.uk/test_images/test.jpg</query>

   <startRecord xsi:type="xsd:int">-1</startRecord>

   <maximumRecords xsi:type="xsd:int">3</maximumRecords>

   <recordSchema xsi:nil="true"/>

   <resultSetId xsi:nil="true"/>

   <sessionId xsi:nil="true"/>

   <sortKey xsi:nil="true"/>

  </ns1:searchRetrieve>

 </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

Figure 5-2 Example Request to the ARTISTE SRW

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

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

 <SOAP-ENV:Body>

  <ns1:searchRetrieveResponse SOAP-
ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns1="SrwService">

   <searchRetrieveResult xsi:type="ns2:Element"
xmlns:ns2="http://xml.apache.org/xml-soap">

    <resultSet>

     <resultSetId>654</resultSetId>

     <numberOfRecords>7</numberOfRecords>

     <records>

      <record>

       <recordSchema>oai_dc</recordSchema>

       <recordData>

        <VisibleLightImage>http://huxelrebe.it-
innovation.soton.ac.uk/servlet/RetrieveImageServlet?imageId=39011&amp;siteId=Sample
</VisibleLightImage>

        <DublinCoreOtherContributors>taylorc</DublinCoreOtherContributors>

        <DublinCoreSubject>WEDDING DRESS</DublinCoreSubject>

        <DublinCoreSource>CT41072</DublinCoreSource>

        <DublinCoreCreator>Posy detail from a Gina Fratini wedding dress comprising
of a cream silk smocked organza dress trimmed with 19th century lace, plus a cap
and posyGiven by Miss Gina Fratini.c.1970</DublinCoreCreator>

        <DublinCoreTitle>Posy detail from a Gina Fratini wedding dress comprising
of a cream silk smocked organza dress trimmed with 19th century lace, plus a cap
and posyGiven by Miss Gina Fratini.c.1970</DublinCoreTitle>

        <DublinCoreIdentifier>pcd658116310757-068</DublinCoreIdentifier>

</recordData>

</record>

        <record>

         <recordSchema>oai_dc</recordSchema>

         <recordData>

          <VisibleLightImage>http://huxelrebe.it-
innovation.soton.ac.uk/servlet/RetrieveImageServlet?imageId=39012&amp;siteId=Sample
</VisibleLightImage>

          <DublinCoreOtherContributors>taylorc</DublinCoreOtherContributors>

          <DublinCoreSubject>WEDDING DRESS</DublinCoreSubject>

          <DublinCoreSource>CT41073</DublinCoreSource>

          <DublinCoreCreator>Cap detail from a Gina Fratini wedding dress
comprising of a cream silk smocked organza dress trimmed with 19th century lace,
plus a cap and posy. Given by Miss Gina Fratini.c.1970</DublinCoreCreator>

          <DublinCoreTitle>Cap detail from a Gina Fratini wedding dress comprising
```

Figure 5-3Example Response to a request to the ARTISTE SRW

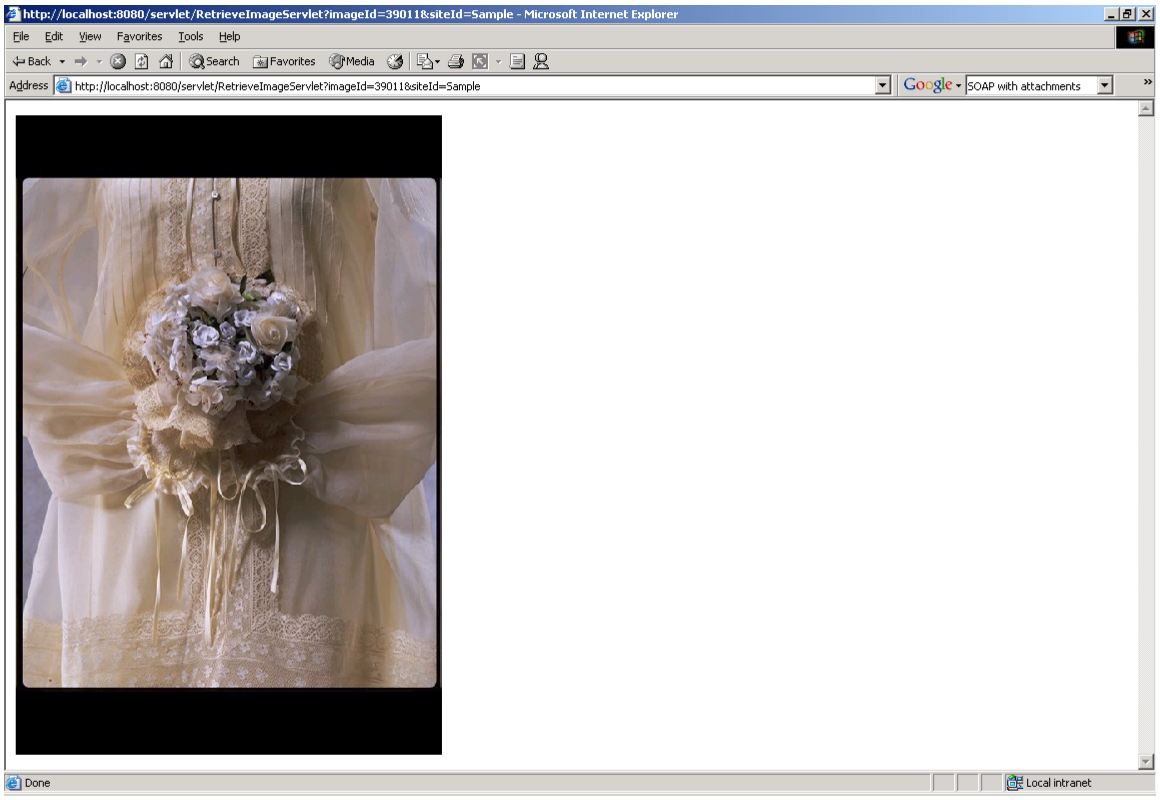

http://localhost:8080/servlet/RetrieveImageServlet?imageId=39011&siteId=Sample - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost:8080/servlet/RetrieveImageServlet?imageId=39011&siteId=Sample
Google SOAP with attachments
Done
Local intranet

Figure 5-4 Image retrieved as part of SRW Record.

6. Impact on Standards

The ARTISTE approach is to use existing standards and technologies where possible for metadata structuring and translation. This underpins an open standards approach to providing an open interface for metadata harvesting and image search and retrieval.

Throughout the project, ARTISTE as tracked appropriate standards and has provided feedback to the community of users of those standards on how the standards could be improved to support images as well as textual information.

More information on ARTISTE impact on international standards for metadata and information retrieval can be found in ARTISTE Deliverable 6.2 [xxiii] which also includes a series of high level observations and recommendations concerning digital library standards.

7. Conclusion

The exploitation of cultural image collections is limited because of a lack of relevant metadata to describe the images, lack of conformance to common schema for metadata that does exist, and lack of appropriate and convenient access methods. The ARTISTE project has addressed these issues not simply by providing access through a single application but the incorporating the ability to deal with content based searching into international standards of information retrieval.

ARTISTE aimed to increase the interoperability of diverse, distributed collections and thus make it possible to use a single interface to, quickly and transparently, search and browse the wealth of digital image information the collections contain as if they were a single entity. The approach we have taken in ARTISTE is to use existing standards and technologies where possible for metadata structuring and translation. ARTISTE makes considerable use of existing open metadata standards such as Dublin Core and RDF Schema. ARTISTE also uses the framework of RDF to enable multilingual access to metadata and metadata thesauri thus addressing another key interoperability issue.

The standards based approach to improving metadata interoperability and access in turn underpins an open standards approach to providing an open interface for metadata harvesting and image search and retrieval. Access to metadata is supported through the Open Archive Initiative (OAI) information retrieval standard for distributed access. A z39.50 based approach for a Search and Retrieve Web service (SRW) is used to provide open query and retrieval of images across multiple collections.

Therefore the image collections in an ARTISTE system are not only interoperable with the other ARTISTE image collections, through the ARTISTE system they also become accessible to many cultural heritage systems and digital library repositories which support the same standards and image retrieval protocols.

8. References

[1] 1 www.artisteweb.org

[2] 1 Addis, M., Lewis, P., Martinez, K. "ARTISTE image retrieval system puts European galleries in the picture", Cultivate Interactive, issue 7, 11 July 2002 http://www.cultivate-int.org/issue7/artiste/

[3] 1 www.sculpteurweb.org

[4] 1 ARTISTE Deliverable D8.3 System Integration, IT Innovation Centre, 2002.

[5] 1 A. Michard, V. Christophides, M. Scholl, M. Stapleton, D. Sutcliffe, A.M. Vercoustre, "The Aquarelle resource discovery system", *Computer Networks and ISDN Systems*, Vol. 30, 1185-1200, 1998.

[6] 1 Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999, http://www.w3.org/TR/REC-rdf-syntax/

[7] 1 Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 27 March 2000, http://www.w3.org/TR/rdf-schema/

[8] 1 W3C RDF Validation Service http://www.w3.org/RDF/Validator/

[9] 1 Dublin Core Metadata Element Set, Version 1.1: Reference Description

[10] http://dublincore.org/documents/dces/

[11] 1 Dublin Core. RDF Schema declaration for the Dublin Core Element Set 1.1

[12] http://purl.org/dc/elements/1.1/

[13] 1 Yves Lafon, Bert Bos, Describing and retrieving photos using RDF and HTTP, 2002 http://www.w3.org/TR/photo-rdf/

[14] 1 SiRPAC RDF Parser http://www-db.stanford.edu/~melnik/rdf/api.html

[15] 1 IEFT RFC 1766 Tags for the Identification of Languages http://www.ietf.org/rfc/rfc1766.txt?number=1766

[16] 1 Xmethods BabelFish Web Service http://www.xmethods.com/ve2/ViewListing.po;jsessionid=NLV1EophdnjdkrYScwfNuTWI(QhxieSRM)?serviceid=14

[17] 1 CERES and National Biological Information Infrastructure (NBII) Biological Resources Division (BRD) Draft RDF Server Standard Documentation http://ceres.ca.gov/thesaurus/RDF.html

[18] 1 XML Writer v.02 is a Java utility class written by Dave Megginson which extends the Java the XMLFilterImpl class(http://www.megginson.com/Software/)

[19] 1 XML Schema for Validating Responses to OAI-PMH Requests http://www.openarchives.org/OAI/openarchivesprotocol.html#OAIPMHschema

[20] 1 National Information Standards Organisation, Z39.50 Information Retrieval Protocol http://www.niso.org/z3950.html, 1998.

[21] 1 Knowledge Integration http://www.k-int.com/jzkit

[22] 1 ARTISTE is an early implementer of the SRW protocol http://www.loc.gov/z3950/agency/zing/srwu/implementors.html

[23] 1 Tomcat http://jakarta.apache.org/tomcat/

[24] 1 Apache http://httpd.apache.org/

[25] 1 "SOAP Messages with Attachments" John J. Barton, Hewlett Packard Labs

[26] Satish Thatte, Microsoft, Henrik Frystyk Nielsen, Microsoft http://www.w3.org/TR/SOAP-attachments

[27] 1 "Impact on World-wide Metadata Standards", ARTISTE Deliverable D6.2, IT Innovation Centre

Appendix A. ARTISTE Core RDF Schema

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

- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/TR/rdf-schema#"
xmlns:ac="http://artiste.it-
innovation.soton.ac.uk/rdf/ArtisteCore.rdf#">

- <!-- -  ARTISTE SCHEMA

  -->

- <!-- -  QueryItem

  -->

- <rdfs:Class rdf:ID="QueryItem">

  <rdfs:subClassOf rdf:resource="http://www.w3.org/1999/02/22-rdf-
syntax#Property" />

  </rdfs:Class>

- <rdfs:Class rdf:ID="Image">

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

  </rdfs:Class>

- <rdfs:Class rdf:ID="Colour">

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

  </rdfs:Class>

- <rdfs:Class rdf:ID="Attribute">

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

  </rdfs:Class>

- <rdfs:Class rdf:ID="TextAttribute">

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

  </rdfs:Class>

- <rdfs:Class rdf:ID="IntegerAttribute">
```

```
  <rdfs:subClassOf rdf:resource="#Attribute" />
  </rdfs:Class>
- <rdfs:Class rdf:ID="RealAttribute">
  <rdfs:subClassOf rdf:resource="#Attribute" />
  </rdfs:Class>
- <rdfs:Class rdf:ID="DateAttribute">
  <rdfs:subClassOf rdf:resource="#Attribute" />
  </rdfs:Class>
- <rdfs:Class rdf:ID="FeatureVectorAttribute">
  <rdfs:subClassOf rdf:resource="#Attribute" />
  </rdfs:Class>
  <rdfs:Property rdf:ID="WholeImage" />
- <rdfs:Property rdf:ID="VisibleLightImage">
  <rdfs:label xml:lang="en">en:Visible Light Image</rdfs:label>
  </rdfs:Property>
- <rdfs:Property rdf:ID="FeatureVector">
  <rdfs:label xml:lang="en">en:FeatureVector</rdfs:label>
  </rdfs:Property>
- <rdfs:Property rdf:ID="SubImage">
  </rdfs:Property>
- <rdfs:Property rdf:ID="PrincipalColour">
  </rdfs:Property>
- <!-- -  QueryOperator
  -->
- <rdfs:Class rdf:ID="QueryOperator">
  </rdfs:Class>
- <rdfs:Class rdf:ID="SimilarTo">
  <rdfs:subClassOf rdf:resource="#QueryOperator" />
  </rdfs:Class>
- <rdfs:Class rdf:ID="PartOf">
  <rdfs:subClassOf rdf:resource="#QueryOperator" />
  </rdfs:Class>
```

```
- <rdfs:Class rdf:ID="Equals">
  <rdfs:subClassOf rdf:resource="#QueryOperator" />
  </rdfs:Class>
- <rdfs:Class rdf:ID="Like">
  <rdfs:subClassOf rdf:resource="#QueryOperator" />
  </rdfs:Class>
- <rdfs:Class rdf:ID="LessThan">
  <rdfs:subClassOf rdf:resource="#QueryOperator" />
  </rdfs:Class>
- <rdfs:Class rdf:ID="GreaterThan">
  <rdfs:subClassOf rdf:resource="#QueryOperator" />
  </rdfs:Class>
- <rdf:Property rdf:ID="AllowedSubject">
  <rdfs:domain rdf:resource="#QueryOperator" />
  <rdfs:range rdf:resource="#QueryItem" />
  </rdf:Property>
- <rdf:Property rdf:ID="AllowedObject">
  <rdfs:domain rdf:resource="#QueryOperator" />
  <rdfs:range rdf:resource="#QueryItem" />
  </rdf:Property>
- <rdfs:Class rdf:ID="QueryExpressionRule">
  </rdfs:Class>
- <ac:QueryExpressionRule rdf:ID="SimilarImage">
  <ac:QueryOperator>#SimilarTo</ac:QueryOperator>
  <ac:AllowedSubject>#Image</ac:AllowedSubject>
  <ac:AllowedObject>#Image</ac:AllowedObject>
  </ac:QueryExpressionRule>
- <ac:QueryExpressionRule rdf:ID="SimilarColour">
  <ac:QueryOperator>#SimilarTo</ac:QueryOperator>
  <ac:AllowedSubject>#Colour</ac:AllowedSubject>
  <ac:AllowedObject>#Colour</ac:AllowedObject>
  </ac:QueryExpressionRule>
```

```
- <ac:QueryExpressionRule rdf:ID="SimilarSubImage">
  <ac:QueryOperator>#PartOf</ac:QueryOperator>
  <ac:AllowedSubject>#Image</ac:AllowedSubject>
  <ac:AllowedObject>#Image</ac:AllowedObject>
  </ac:QueryExpressionRule>
- <ac:QueryExpressionRule rdf:ID="SimilarFeatureVector">
  <ac:QueryOperator>#SimilarTo</ac:QueryOperator>
  <ac:AllowedSubject>#FeatureVectorAttribute</ac:AllowedSubject>
  <ac:AllowedObject>#FeatureVectorAttribute</ac:AllowedObject>
  </ac:QueryExpressionRule>
- <ac:QueryExpressionRule rdf:ID="TextEquals">
  <ac:QueryOperator>#Equals</ac:QueryOperator>
  <ac:AllowedSubject>#TextAttribute</ac:AllowedSubject>
  <ac:AllowedObject>#TextAttribute</ac:AllowedObject>
  </ac:QueryExpressionRule>
- <ac:QueryExpressionRule rdf:ID="TextLike">
  <ac:QueryOperator>#Like</ac:QueryOperator>
  <ac:AllowedSubject>#TextAttribute</ac:AllowedSubject>
  <ac:AllowedObject>#TextAttribute</ac:AllowedObject>
  </ac:QueryExpressionRule>
- <ac:QueryExpressionRule rdf:ID="RealEquals">
  <ac:QueryOperator>#Equals</ac:QueryOperator>
  <ac:AllowedSubject>#RealAttribute</ac:AllowedSubject>
  <ac:AllowedObject>#RealAttribute</ac:AllowedObject>
  </ac:QueryExpressionRule>
- <ac:QueryExpressionRule rdf:ID="RealLessThan">
  <ac:QueryOperator>#LessThan</ac:QueryOperator>
  <ac:AllowedSubject>#RealAttribute</ac:AllowedSubject>
  <ac:AllowedObject>#RealAttribute</ac:AllowedObject>
  </ac:QueryExpressionRule>
- <ac:QueryExpressionRule rdf:ID="RealGreaterThan">
  <ac:QueryOperator>#GreaterThan</ac:QueryOperator>
  <ac:AllowedSubject>#RealAttribute</ac:AllowedSubject>

  <ac:AllowedObject>#RealAttribute</ac:AllowedObject>

  </ac:QueryExpressionRule>

- <ac:QueryExpressionRule rdf:ID="IntegerEquals">

  <ac:QueryOperator>#Equals</ac:QueryOperator>

  <ac:AllowedSubject>#IntegerAttribute</ac:AllowedSubject>

  <ac:AllowedObject>#IntegerAttribute</ac:AllowedObject>

  </ac:QueryExpressionRule>

- <ac:QueryExpressionRule rdf:ID="IntegerLessThan">

  <ac:QueryOperator>#LessThan</ac:QueryOperator>

  <ac:AllowedSubject>#IntegerAttribute</ac:AllowedSubject>

  <ac:AllowedObject>#IntegerAttribute</ac:AllowedObject>

  </ac:QueryExpressionRule>

- <ac:QueryExpressionRule rdf:ID="IntegerGreaterThan">

  <ac:QueryOperator>#GreaterThan</ac:QueryOperator>

  <ac:AllowedSubject>#IntegerAttribute</ac:AllowedSubject>

  <ac:AllowedObject>#IntegerAttribute</ac:AllowedObject>

  </ac:QueryExpressionRule>

- <!-- -  Analyser

  -->

- <rdfs:Class rdf:ID="Analyser">

  <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/rdf-
schema#Resource" />

  </rdfs:Class>

- <rdf:Property rdf:ID="AnalyserAppliesTo">

  <rdfs:domain rdf:resource="#Analyser" />

  <rdfs:range rdf:resource="#QueryExpressionRule" />

  </rdf:Property>

- <!--  Schema

  -->

- <rdfs:Class rdf:ID="Schema">

  </rdfs:Class>

- <rdfs:Class rdf:ID="Thesaurus">
```

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

  <rdfs:isDefinedBy rdf:resource="http://artiste.it-
innovation.soton.ac.uk/rdf/thesaurus.rdf" />

  </rdfs:Class>

- <rdfs:Class rdf:ID="MetadataSchema">

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

  <rdfs:isDefinedBy rdf:resource="http://artiste.it-
innovation.soton.ac.uk/rdf/ArtisteCore.rdf#MetadataSchema" />

  </rdfs:Class>

- <!-- -  General classes and properties

  -->

- <rdfs:Class rdf:ID="URI">

  <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/rdf-
schema#Resource" />

  </rdfs:Class>

- <rdf:Property rdf:ID="Name">

  </rdf:Property>

- <rdf:Property rdf:ID="Description">

  </rdf:Property>

- <rdf:Property rdf:ID="Hostname">

  </rdf:Property>

- <rdf:Property rdf:ID="Port">

  </rdf:Property>

- <!-- -  Site

  -->

- <rdfs:Class rdf:ID="Site">

  <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/rdf-
schema#Resource" />

  </rdfs:Class>

- <rdf:Property rdf:ID="SiteName">

  <rdfs:subPropertyOf rdf:resource="#Name" />

  <rdfs:domain rdf:resource="#Site" />

  <rdfs:range rdf:resource="http://www.w3.org/TR/rdf-
schema#Literal" />

  </rdf:Property>
```

```
- <rdf:Property rdf:ID="SiteDescription">
  <rdfs:subPropertyOf rdf:resource="#Description" />
  <rdfs:domain rdf:resource="#Site" />
  <rdfs:range rdf:resource="http://www.w3.org/TR/rdf-
schema#Literal" />
  </rdf:Property>
- <rdf:Property rdf:ID="SiteSchemaLocation">
  <rdfs:domain rdf:resource="#Site" />
  <rdfs:range rdf:resource="#URI" />
  </rdf:Property>
  </rdf:RDF>
```

Appendix B. Dublin Core RDF Schema

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

- <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/TR/rdf-schema#"
xmlns:dc="http://artiste.it-
innovation.soton.ac.uk/test_rdf/dc.rdf#">

- <rdf:Description rdf:about="">

  <dc:title>The Dublin Core Element Set</dc:title>

  <dc:creator>The Dublin Core Metadata Inititative</dc:creator>

  <dc:description>The Dublin Core is a simple metadata element set
intended to facilitate discovery of electronic
resources.</dc:description>

  <dc:date>1995-03-01</dc:date>

  </rdf:Description>

- <rdf:Description ID="title">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />

  <rdfs:label xml:lang="en">en:Dublin Core Title</rdfs:label>

  <rdfs:label xml:lang="fr">fr:Dublin Core Titre</rdfs:label>

  <rdfs:label xml:lang="it">it:Dublin Core Titolo</rdfs:label>

  <rdfs:comment xml:lang="en">en:The name given to the resource,
usually by the Creator or Publisher.</rdfs:comment>

  <rdfs:comment xml:lang="fr">fr:Typiquement, un titre sera le nom
par lequel la ressource est officiellement connue.</rdfs:comment>

  <rdfs:comment xml:lang="it">it:Un nome dato alla
risorsa</rdfs:comment>

  </rdf:Description>

- <rdf:Description ID="creator">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />

  <rdfs:label xml:lang="en">en:Dublin Core Creator</rdfs:label>
  <rdfs:label xml:lang="fr">fr:Dublin Core Createur</rdfs:label>

  <rdfs:label xml:lang="it">it:Dublin Core Creatore</rdfs:label>

  <rdfs:comment xml:lang="en">en:The person or organization
primarily responsible for creating the intellectual content of the
resource. For example, authors in the case of written documents,
artists, photographers, or illustrators in the case of visual
resources.</rdfs:comment>

  <rdfs:comment xml:lang="fr">fr:Exemples de Createur incluent une
personne, une organisation, ou un service. Typiquement, un nom du
Createur devrait etre utilise pour designer cette
entite.</rdfs:comment>

  <rdfs:comment xml:lang="it">it:L argomento della
risorsa.</rdfs:comment>

  </rdf:Description>

- <rdf:Description ID="subject">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />

  <rdfs:label xml:lang="en">en:Dublin Core Subject</rdfs:label>

  <rdfs:label xml:lang="fr">fr:Dublin Core Sujet et mots-
clefs</rdfs:label>

  <rdfs:label xml:lang="it">it:Dublin Core Soggetto e Parole
chiave</rdfs:label>

  <rdfs:comment xml:lang="en">en:The topic of the resource.
Typically, subject will be expressed as keywords or phrases that
describe the subject or content of the resource. The use of
controlled vocabularies and formal classification schemes is
encouraged.</rdfs:comment>

  <rdfs:comment xml:lang="fr">fr:Le sujet du contenu de la
ressource.</rdfs:comment>

  <rdfs:comment xml:lang="it">it:L argomento della
risorsa.</rdfs:comment>

  </rdf:Description>

- <rdf:Description ID="description">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />

  <rdfs:label xml:lang="en">en:Dublin Core
Description</rdfs:label>

  <rdfs:label xml:lang="fr">fr:Dublin Core
Description</rdfs:label>

  <rdfs:label xml:lang="it">it:Dublin Core Titolo</rdfs:label>

  <rdfs:comment xml:lang="en">en: A textual description of the
content of the resource, including abstracts in the case of
```

```
document-like objects or content descriptions in the case of
visual resources.</rdfs:comment>

  <rdfs:comment xml:lang="fr">fr:Une description du contenu de la
ressource.</rdfs:comment>

  <rdfs:comment xml:lang="it">it:Una spiegazione del contenuto
della risorsa</rdfs:comment>

  </rdf:Description>

- <rdf:Description ID="publisher">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />

  <rdfs:label xml:lang="en">en:Dublin Core Publisher</rdfs:label>

  <rdfs:comment xml:lang="en">en:The entity responsible for making
the resource available in its present form, such as a publishing
house, a university department, or a corporate
entity.</rdfs:comment>

  </rdf:Description>

- <rdf:Description ID="contributor">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />

  <rdfs:label xml:lang="en">en:Dublin Core Other
Contributors</rdfs:label>

  <rdfs:comment xml:lang="en">en:A person or organization not
specified in a Creator element who has made significant
intellectual contributions to the resource but whose contribution
is secondary to any person or organization specified in a Creator
element (for example, editor, transcriber, and
illustrator).</rdfs:comment>

  </rdf:Description>

- <rdf:Description ID="type">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />

  <rdfs:label xml:lang="en">en:Dublin Core Type</rdfs:label>

  <rdfs:comment xml:lang="en">en:The category of the resource,
such as home page, novel, poem, working paper, technical report,
essay, dictionary. For the sake of interoperability, Type should
be selected from an enumerated list that is currently under
development in the workshop series.</rdfs:comment>

  </rdf:Description>

- <rdf:Description ID="format">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />
```

```
  <rdfs:label xml:lang="en">en:Dublin Core Format</rdfs:label>

  <rdfs:comment xml:lang="en">en:The data format of the resource,
used to identify the software and possibly hardware that might be
needed to display or operate the resource. For the sake of
interoperability, Format should be selected from an enumerated
list that is currently under development in the workshop
series.</rdfs:comment>

  </rdf:Description>

- <rdf:Description ID="identifier">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />

  <rdfs:label xml:lang="en">en:Dublin Core Identifier</rdfs:label>

  <rdfs:comment xml:lang="en">en:A string or number used to
uniquely identify the resource. Examples for networked resources
include URLs and URNs (when implemented). Other globally-unique
identifiers, such as International Standard Book Numbers (ISBN) or
other formal names are also candidates for this
element.</rdfs:comment>

  </rdf:Description>

- <rdf:Description ID="source">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />

  <rdfs:label xml:lang="en">en:Dublin Core Source</rdfs:label>

  <rdfs:comment xml:lang="en">en:Information about a second
resource from which the present resource is derived. While it is
generally recommended that elements contain information about the
present resource only, this element may contain a date, creator,
format, identifier, or other metadata for the second resource when
it is considered important for discovery of the present resource;
recommended best practice is to use the Relation element instead.
For example, it is possible to use a Source date of 1603 in a
description of a 1996 film adaptation of a Shakespearean play, but
it is preferred instead to use Relation "IsBasedOn" with a
reference to a separate resource whose description contains a Date
of 1603. Source is not applicable if the present resource is in
its original form.</rdfs:comment>

  </rdf:Description>

- <rdf:Description ID="language">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />

  <rdfs:label xml:lang="en">en:Dublin Core Language</rdfs:label>

  <rdfs:comment xml:lang="en">en:The language of the intellectual
content of the resource. Where practical, the content of this
field should coincide with RFC 1766 [Tags for the Identification
of Languages, http://ds.internic.net/rfc/rfc1766.txt ]; examples
include en, de, es, fi, fr, ja, th, and zh.</rdfs:comment>

  </rdf:Description>

- <rdf:Description ID="relation">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />

  <rdfs:label xml:lang="en">en:Dublin Core Relation</rdfs:label>

  <rdfs:comment xml:lang="en">en:An identifier of a second
resource and its relationship to the present resource. This
element permits links between related resources and resource
descriptions to be indicated. Examples include an edition of a
work (IsVersionOf), a translation of a work (IsBasedOn), a chapter
of a book (IsPartOf), and a mechanical transformation of a dataset
into an image (IsFormatOf). For the sake of interoperability,
relationships should be selected from an enumerated list that is
currently under development in the workshop series.</rdfs:comment>

  </rdf:Description>

- <rdf:Description ID="coverage">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />

  <rdfs:label xml:lang="en">en:Dublin Core Coverage</rdfs:label>

  <rdfs:comment xml:lang="en">en:The spatial or temporal
characteristics of the intellectual content of the resource.
Spatial coverage refers to a physical region (e.g., celestial
sector); use coordinates (e.g., longitude and latitude) or place
names that are from a controlled list or are fully spelled out.
Temporal coverage refers to what the resource is about rather than
when it was created or made available (the latter belonging in the
Date element); use the same date/time format (often a range) [Date
and Time Formats (based on ISO8601), W3C Technical Note,
http://www.w3.org/TR/NOTE-datetime] as recommended for the Date
element or time periods that are from a controlled list or are
fully spelled out.</rdfs:comment>

  </rdf:Description>

- <rdf:Description ID="rights">

  <rdf:type rdf:resource="http://www.w3.org/TR/REC-rdf-
syntax#Property" />

  <rdfs:label xml:lang="en">en:Dublin Core Rights</rdfs:label>

  <rdfs:comment xml:lang="en">en:A rights management statement, an
identifier that links to a rights management statement, or an
identifier that links to a service providing information about
rights management for the resource.</rdfs:comment>

  </rdf:Description>

  </rdf:RDF>
```

Appendix C. BNF for CAQL Syntax

```
caql-query ::=base-expre *(“and” base-expr)

base-expr::=primary | caql-query

primary ::=result-set-expression | [index-name rel-op] adj-
expr | index-name img-op img-analyser img-exp

result-set-expression ::=”resultSet =” identifier

index-name ::= [index-prefix “.”]index-base-name

index-base-name ::= identifier

index-prefix ::=identifier

rel-op ::=”=”|”contains”

img-op ::= “SimilarTo” | “PartOf”

img-analyser ::= identifier

img-expr ::= url

adj-expr ::=term *(“and” adj-expr)

term ::=identifier|quoted-string-literal
```

Identifiers and Literals
An identifier is a sequence of letters, digits, and underscores. Quoted string literals are required for terms containing any other characters, and start with a double-quote and end with a matching closing double-quote. To allow double-quotes to be embedded in a string, two double-quotes in a row generate a single literal double-quotes character without terminating the string literal

URL

protocol://hostname/pathname

Reserved Words
The following are reserved words and so must be quoted if used as text in a term. As reserved words they are case-insensitive.

and

resultSet

SimilarTo

Part of

Contains

---

i www.artisteweb.org
ii Addis, M., Lewis, P., Martinez, K. "ARTISTE image retrieval system puts European galleries in the picture", Cultivate Interactive, issue 7, 11 July 2002 http://www.cultivate-int.org/issue7/artiste/
iii www.sculpteurweb.org
iv ARTISTE Deliverable D8.3 System Integration, IT Innovation Centre, 2002.
v A. Michard, V. Christophides, M. Scholl, M. Stapleton, D. Sutcliffe, A.M. Vercoustre, "The Aquarelle resource discovery system", *Computer Networks and ISDN Systems*, Vol. 30, 1185-1200, 1998.
vi Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation 22 February 1999, http://www.w3.org/TR/REC-rdf-syntax/
vii Resource Description Framework (RDF) Schema Specification 1.0, W3C Candidate Recommendation 27 March 2000, http://www.w3.org/TR/rdf-schema/
viii Dublin Core Metadata Element Set, Version 1.1: Reference Description http://dublincore.org/documents/dces/
ix Dublin Core. RDF Schema declaration for the Dublin Core Element Set 1.1 http://purl.org/dc/elements/1.1/
x Yves Lafon, Bert Bos, Describing and retrieving photos using RDF and HTTP, 2002 http://www.w3.org/TR/photo-rdf/
xi SiRPAC RDF Parser http://www-db.stanford.edu/~melnik/rdf/api.html
xii IEFT RFC 1766 Tags for the Identification of Languages http://www.ietf.org/rfc/rfc1766.txt?number=1766
xiii Xmethods BabelFish Web Service http://www.xmethods.com/ve2/ViewListing.po;jsessionid=NLV1EophdnjdkrYScwfNuTWI(QhxieSRM)?serviceid=14
xiv CERES and National Biological Information Infrastructure (NBII) Biological Resources Division (BRD) Draft RDF Server Standard Documentation http://ceres.ca.gov/thesaurus/RDF.html
xv XML Writer v.02 is a Java utility class written by Dave Megginson which extends the Java the XMLFilterImpl class(http://www.megginson.com/Software/)

xvi XML Schema for Validating Responses to OAI-PMH Requests
http://www.openarchives.org/OAI/openarchivesprotocol.html#OAIPMHschema
xvii National Information Standards Organisation, Z39.50 Information Retrieval Protocol http://www.niso.org/z3950.html, 1998.
xviii Knowledge Integration http://www.k-int.com/jzkit
xix ARTISTE is an early implementor of the SRW protocol
http://www.loc.gov/z3950/agency/zing/srwu/implementors.html
xx Tomcat http://jakarta.apache.org/tomcat/
xxi Apache http://httpd.apache.org/
xxii "SOAP Messages with Attachments" John J. Barton, Hewlett Packard Labs Satish Thatte, Microsoft, Henrik Frystyk Nielsen, Microsoft
http://www.w3.org/TR/SOAP-attachments
xxiii "Impact on World-wide Metadata Standards", ARTISTE Deliverable D6.2, IT Innovation Centre