Implementing a Semantic Catalogue of Geospatial Data

Helbert Arenas, Benjamin Harbelot and Christophe Cruz
Laboratoire Le2i, UMR-6302 CNRS,Département d'Informatique , Université de Bourgogne, 7 Boulevard Docteur Petitjean, 21078 Dijon, France,
{*helbert.arenas,benjamin.harbelot*}*@checksem.fr, christophe.cruz@u-bourgogne.fr*

Keywords: CSW, OGC, triplestore, metadata

Abstract: Complex spatial analysis requires the combination of heterogeneous datasets. However the identification of a dataset of interest is not a trivial task. Users need to review metadata records in order to select the most suitable datasets. We propose the implementation of a system for metadata management based on semantic web technologies.Our implementation helps the user with the selection task. In this paper we present a CSW that uses a triplestore as its metadata repository. We implement a translator between Filter Encoding and SPARQL/GeoSPARQL in order to comply to basic OGC standards. Our results are promising however, this is a novel field with room for improvement.

1 INTRODUCTION

There is a growing interest in the development of the SDI (Spatial Data Infrastructure), a term that refers to the sharing of information and resources between different institutions.The term was first used by the United States National Research Council in 1993. It refers to the set of technologies, policies and agreements designed to allow the communication between spatial data providers and users (ESRI, 2010).

Currently vast amounts of information are being deployed in the internet through web services. In the spatial domain this has been possible, thanks, in a significant part to the standardization efforts by OGC (Open Geospatial Consortium). OGC is an international industry and academic group whose goal is to develop open standards that enable communication between heterogeneous systems (OGC, 2012).

By being able to combine diverse spatial data sources, researchers and decision makers would be able to implement *smart queries*. This is a term first employed by Goodwin (2005). It refers to the combination of heterogeneous data sources in order to solve complex problems (Goodwin, 2005). The first step towards implementing *smart queries*, is to allow users to identify the most suitable dataset.

The tasks in which OGC is interested are: publishing, finding and binding spatial information. OGC provides standards that allow data providers and users to communicate using a common language. The data is offered through web services such as WFS (Web Feature Service), WMS (Web Map Service) or SOS (Sensor Observation Service). However in order to identify a dataset of interest a user needs first to search among the different offered datasets by using a CSW (Catalogue Service for the Web).

OGC defines the interfaces and operations to query metadata records. There are both commercial and opensource/freeware CSW implementations. Among the commercials we can find ESRI ArcGIS server and MapInfo Manager. Among the opensource implementations we find Constellation, Degree and GeoNetworkCSW. The OGC standards do not indicate specific software components. In the case of CSW, developers are able to select the metadata repository more suitable to their preferences/requirements. However, the OGC CSW standard indicates the metadata formats that should be supported. There are several metadata standards currently in use: ISO 19115 (for geographic information), ISO19119 (for services), ISO 19139 (metadata XML schema implementation), Dublin Core (ISO 15836) or the FGDC content standard for digital geospatial metadata. Traditional configuration of a CSW uses a relational database as the metadata records repository. That is the case of GeoNetwork, arguably one of the most popular CSW implementations deployed in the web. GeoNetwork by default uses a McKoiDB relational database, although it can connect to MySQL, PostGreSQL and other RDBMS (Dunne et al., 2012). In order to populate a metadata repository the CSW manager can use a meta-

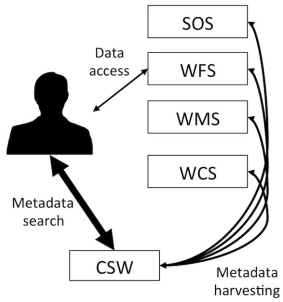

Figure 1: OGC services.

data editor program or run a metadata *harvest* application. Queries submitted to a CSW should be formatted as Filter Encoding or as CQL. The former is a XML encoded query language, while the later is a human readable text encoded query language (OSGeo, 2012)(Vretanos, 2005).

Currently in most implementations the metadata repository is a relational database. Therefore, queries are performed by matching strings to selected metadata elements. In this paper we propose the use of semantic web technologies to store and query metadata records. By using these technologies we are able to take advantage of inference and reasoning mechanisms not available on relational databases. In Section 2 we review research conducted by other teams in the same field. In Section 3 we describe how we have implemented our model. Finally in Section 4 we present our conclusions and outline future research.

2 RELATED RESEARCH

To implement a *smart query* a user must be able to identify relevant datasets and access them. The process implemented within SDI with OGC services is depicted in figure 1. The spatial data is offered by different providers through web services with implementations such as WCS, SOS, WFS or WMS. These services might belong to different institutions, with servers located in multiple countries, be implemented by different vendors, or be deployed using different languages. However, by implementing OGC standards all of them have common request and response contents, parameters and encodings. These common elements allow a user to access different services using a proven, safe strategy . In order to allow datasets to be *discoverable*, they have to be published in a catalogue service CSW. The metadata for the datasets is obtained by the catalogue service using a *harvesting* operation. The user in order to discover a specific dataset, submits a query. The server processes the query using a string matching process, and sends a response to the user. Once the user has identified the relevant dataset, she is able to obtain the actual vector/raster data and perform a specific analysis.

The string matching process is a major limitation in the current SDI previously identified by other researchers. In (Kammersell and Dean, 2007) the authors aim to integrate heterogeneous datasources. In this research the authors propose the creation of a layer that translates the users query formulated in OWL to a WFS XML request format. Later, they propose do the inverse process with the results. Another approach is proposed by (Kolas et al., 2005). Here the authors propose the implementation of five different ontologies: 1) Base Geospatial Ontology for basic geospatial concepts resulting from the conversion of GML schemas into OWL. 2) Domain Ontology, this is the users ontology. Its purpose is to link users concepts to the base geospatial ontology. 3) Geospatial Service Ontology, used to describe services and allow discovery. 4) Geospatial Filter Ontology, which is used to formalize filter description and use. 5) Feature Data Source Ontology, to represent the characteristics of the features returned from the WFS. Another approach is described by (Harbelot et al., 2013), here the authors suggest the integration of data from OGC services into a triplestore with a focus on the WFS filters. In (Janowicz et al., 2010) (Janowicz et al., 2012), the authors propose the addition of semantic annotations for each level of a geospatial semantic chain process that involves OGC services. For instance, they propose specific semantic annotations at the level of the service OGC Capabilities document that would correspond to all the datasets managed by the service. Other annotations would correspond to specific data layers. Spatial Data with semantic annotations could later be processed and semantically analysed using custom made reasoning services. To achieve this goal they propose the deployment of OGC services capable of interacting with libraries such as *Sapience* which would result in richer data and data descriptions. However there is little development in this direction. At the moment there is little use of semantic annotations on OGC capabilities documents.

In (Gwenzi, 2010) the author describes the CSW limitations by evaluating GeoNetwork, a popular open source CSW implementation. The author identifies three ways in which it is possible to add semantic annotations to the CSW: 1) By associating keywords to concepts using the *getCapabilities* response. 2) By adding a link in the GeoNetwork client interface to a ontology browser. In this way the user instead of using keywords, would be able to utilize the hierarchical structure to identify the topic that best suits her interest. 3) Adding ontologies as an extension pack-

age using ebRIM. In this work, after considering her alternatives the author choose the third option.

Yue et al. (2006) extend the ebRIM CSW specification by: 1) adding new classes based on existing ebRIM classes; and 2) adding Slots to existing classes, thus creating new attributes. As a result they are able to store richer metadata records in the catalogue. The authors identified two possible options to implement a search functionality: 1) create an external component without further modification of the CSW schemas; 2) modify the CSW adding semantic functionalities to the existing CSW schemas. In this research they choose the first option (Yue et al., 2006). Yue et al. (2011) extends this work, adding further development in the field of geoservices (Yue et al., 2011).

A different approach is used by (Lopez-Pellicer et al., 2010). In this research the goal is to provide access to data stored in CSW as Linked Data. In order to achieve this goal the authors developed CSW2LD, a middle layer on top of a conventional CSW based server. It allows the server to mimic other Linked Data sources and publish metadata records. CSW2LD wraps the following CSW requests: *GetCapabilities*, *GetRecords* and *GetRecordById*.

A very interesting work in progress is described in (Pigot, 2012). This is a website describing a proposal by a team from the GeoNetwork developer community. The authors intend to perform a major change in GeoNetwork, allowing it to store metadata as RDF facts stored in a RDF repository. They intent to use SPARQL/GeoSPARQL to retrieve data. The website describe technical characteristics of GeoNetwork and mentions fields that require work in order to implement the project. Currently queries in GeoNetwork are formatted as Filter Encoding or as CQL. Any implementation of a RDF metadata repository would need to consider a translation mechanism between the current queries format to SPARQL (a W3C recommendation) (DuCharme, 2011). Regarding the spatial component of queries, currently GeoNetwork handles spatial constraints using GeoTools. In the semantic web domain, spatial queries are performed using GeoSPARQL (Kolas and Batle, 2012). According to the authors it is not clear if GeoSPARQL is mature enough to handle metadata spatial queries. Even more there is no mechanism to translate spatial constraints into GeoSPARQL. Regardless of the advantages that semantic web technologies might bring into CSWs there is scarce research on this topic. By the time we wrote this paper, there was no further development in (Pigot, 2012) and the website was last updated by the end of October of 2012.

3 IMPLEMENTATION

In this paper we present a minimalistic implementation of a CSW in which the metadata records are mapped to an ontology. Our CSW is implemented as a Java Servlet. The metadata information is stored in a Parliament triplestore. We opted for Parliament because of its spatial capabilities thanks to its support for GeoSPARQL. Our Servlet is able to respond to *GetRecords* requests submitted as *POST*. Our implementation translates Filter Encoding formatted queries into SPARQL/GeoSPARQL.The response of our servlet follows the *csw:SummaryRecord* format.

Our implementation is a proof of concept is completed, however there is plenty of room for improvement. It uses a geometries dataset that represent toponyms in order to facilitate user quests. Additionally it uses a taxonomy of concepts to show how formal relations between concepts can improve the metadata search. For this paper we are using a dummy taxonomy in order to show potential uses of this approach. In the near future we plan to deploy a real taxonomy and the mechanisms required to determine class membership automatically. Our current work on this topic is based on ideas introduced by (Werner et al., 2012).

In the next subsections we further describe how we obtain our metadata records,map the information to an ontology, and perform queries.

3.1 Harvesting Metadata

We focused our research on metadata records for datasets available on WFSs. To obtain the metadata information we have developed a tool that makes use of the WFS standard requests *GetCapabilities* and *DescribeFeatureType*. The response from the request is later mapped to Dublin core metadata elements and loaded into a triplestore. In total we have harvested information from the 17 WFSs, having as a result 2690 metadata records (See Appendix).

Figure 2 depicts the metadata elements obtained and how they are mapped in the ontology. The circles represent classes in the ontology, while the rectangles depict literal or empty nodes. The coloured boxes represent elements for which we are able to obtain information from WFS responses.

The metadata records are stored as instances of the ontology class *abc:MetadataRecord*, which has properties corresponding to Dublin Core elements. This class is a subclass of *geo:Feature*, therefore has a spatial representation (the bounding box), which allows spatial queries (See Figures 3 and 2).

The following code is fragment of a *GetCapabilities* response, from which we can obtain information

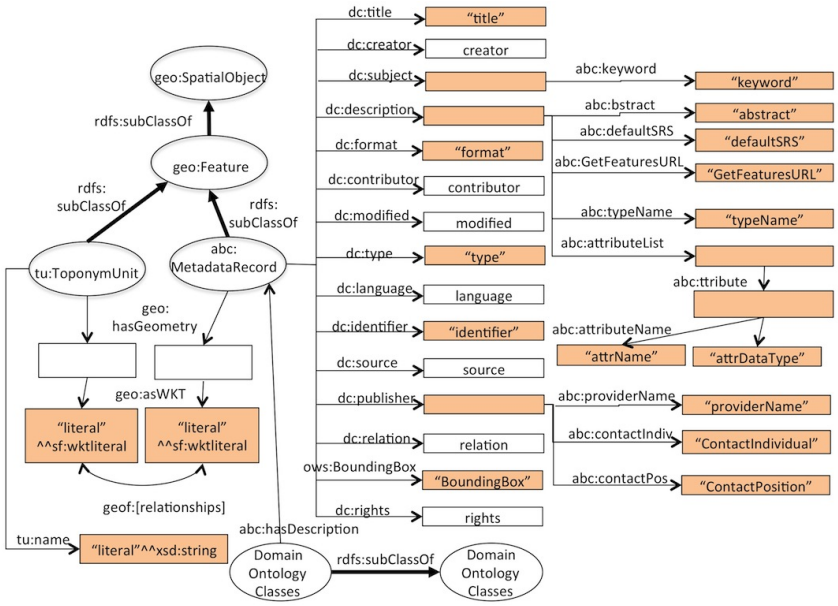

Figure 2: Classes, properties and literals for a metadata record.

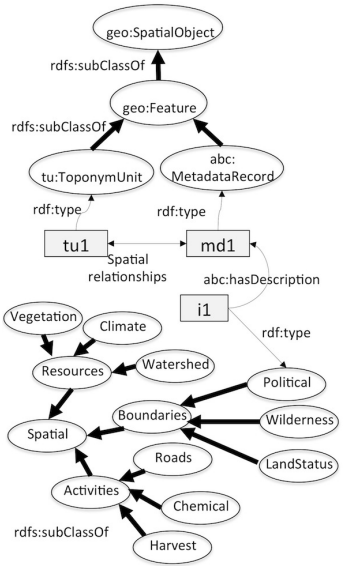

Figure 3: Classes, instances and relationships in the proposed model.

regarding the WFS publishing entity. This information is mapped into the *dc:publisher* Dublin Core element (See figure 2).

```
 <ows:ServiceProvider>
<ows:ProviderName>
Provider X - Fishery department
</ows:ProviderName>
<ows:ServiceContact>
<ows:IndividualName>Helbert<ows:IndividualName/>
<ows:PositionName>GIS Manager<ows:PositionName/>
<ows:ContactInfo>
<ows:Phone>
<ows:Voice>225-568368<ows:Voice/>
<ows:Facsimile/>
</ows:Phone>
<ows:Address>
<ows:City>Roma</ows:City>
<ows:AdministrativeArea>
Bourgogne
</ows:AdministrativeArea>
<ows:PostalCode>00100</ows:PostalCode>
<ows:Country>France</ows:Country>
</ows:Address>
</ows:ContactInfo>
</ows:ServiceContact>
</ows:ServiceProvider>
```

The following XML code, is another part of a *GetCapabilities* response. From this segment of the response, we can obtain information for: *dc:title*, *dc:subject*, *dc:description* and *ows:BoundingBox*.

```
 <FeatureType xmlns:example=
 "http://www.example-provider.org/example">
<Name>example:name</Name>
<Title>Example dataset title</Title>
<Abstract>Example abstract</Abstract>
<ows:Keywords>
<ows:Keyword>example keyword1</ows:Keyword>
</ows:Keywords>
<DefaultSRS>
urn:x-ogc:def:crs:EPSG:4326
</DefaultSRS>
<ows:WGS84BoundingBox>
<ows:LowerCorner>-5.84 37.75</ows:LowerCorner>
<ows:UpperCorner>11.02 54.63</ows:UpperCorner>
</ows:WGS84BoundingBox>
</FeatureType>
```

Additionally, our harvesting tool, submits a *DescribeFeatureType* request for each layer of information found in the WFS. From the response we are able to obtain a list of attributes for the elements in the dataset. The list of attributes is added to the *dc:description* metadata element. The following XML code depicts part of a response to a *DescribeFeatureType* request.

```
<xsd:complexType name="country_boundsType">
<xsd:complexContent>
<xsd:extension base="gml:AbstractFeatureType">
<xsd:sequence>
<xsd:element maxOccurs="1" minOccurs="0"
name="THE_GEOM" nillable="true"
type="gml:MultiSurfacePropertyType"/>
<xsd:element maxOccurs="1" minOccurs="0"
name="AREA" nillable="true"
type="xsd:double"/>
<xsd:element maxOccurs="1" minOccurs="0"
name="STATUS" nillable="true"
type="xsd:string"/>
<xsd:element maxOccurs="1" minOccurs="0"
name="TERR_NAME" nillable="true"
type="xsd:string"/>
</xsd:sequence>
</xsd:extension>
</xsd:complexContent>
</xsd:complexType>
```

3.2 Link to Concept Classification

By using semantic web technologies we are not limited to string matching queries. We can also use inference mechanisms based on subsuming and established relationships between terminology and concepts. To test these capabilities we have implemented a taxonomy with a *domain ontology* classes. The relationships between these concepts are of the type *subclassOf*. For any instance of a *domain ontology* class exist a corresponding instance of the class *abc:MetadataRecord* that describes it. The association between instances of *abc:MetadataRecord* and an instance of any of the *domain ontology* classes is done with the *abc:hasDescription* property. Using the *domain ontology* we can infer instance class membership. Figure 3 depicts our classification. In this case, classes *Resources*, *Boundaries* and *Activities* are subclasses of *Spatial*. Using the ontology we can infer that all instances of the class *Political* are also instances of the class *Boundaries* and the class *Spatial*.

The goal of this paper is to show potential uses of semantic web technologies for metadata record management. For the examples in this paper the domain ontology class membership is assigned randomly. We intent in the future to define mechanisms to automatically identify the class membership with information from the metadata record. Our work in this topic is based on an approach suggested by (Werner et al., 2012).

3.3 Toponym Elements

In order to test the spatial capabilities of the triplestore we define a class *tu:ToponymUnit*.This class is a subclass of *geo:Feature* therefore instances of this class have a spatial representation. Thanks to the ontology we can establish spatial relations between instances of *tu:ToponymUnit* and *abc:MetadataRecord* (See figure 3). This approach helps the user to define a spatial search. It is easier for the user define a search by using a familiar name than by using a set of coordinates.

We obtained a country political boundaries dataset from Esri and DeLorme Publishing Company, Inc. under a Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License (ESRI, 2011). The dataset is a shapefile with 668 features, each feature is a multipolygon. It may be that the complete boundaries of a country result from the aggregation of several features. For instance, in the case of Spain, there is one feature that depicts the Spanish domains in the Iberian peninsula, a second feature for the African enclaves, a third one for the Baleares islands, and so on.

Before uploading our political boundary dataset, we first processed it in order to convert the features to single polygons, then delete polygons that we considered too small for practical purposes. Later we simplify the remaining polygons reducing the number of vertices. Finally we uploaded the dataset using a customized Java program, that uses Jena and GeoTools libraries. The pre-loading processing was done using Quantum GIS and GeoTools. The final result is 3037 instances of the class *tu:ToponymUnit*.

3.4 Metadata Query

In order to test our implementation we deployed a website in our intranet. It uses HTML and JavaScript to allow the users to define queries.

At start, the web site requests the list of domain ontology classes:

```
SELECT DISTINCT
?class
WHERE
{ ?class rdfs:subClassOf xyz:Spatial.}
```

And the list of toponyms in the triplestore:

```
Select Distinct ?tu_name
WHERE
{?c a tu:ToponymUnit.
?c tu:CountryName ?tu_name.}
```

The servlet responds with lists that are stored in the website as arrays, that are used to populate combo boxes in the user interface (See figures 4 6 and 5).

The user interface allows the user to define one or more constraints for the metadata search. A JavaScript running on the client side, formats the query as a XML document following the Filter Encoding specification. Once the XML query arrives to the server, the servlet proceeds to decompose it, into the constituent constraints. Then the servlet proceeds to translate each constraint separately into the respective set of triples and filter elements. Once all the constraints have been translated, the triples and filter elements are merged into a single SPARQL/GeoSPARQL query, which is submitted to the triplestore. The response from the triplestore is then formatted by the servlet as *csw:SummaryRecord* and sent to the client website.

Using the interface the user is able to define three types of constraints:

1. Domain ontology class membership: Each metadata record describes an entity with a class membership. This constraint allows the user to identify the class membership of the entity (See figure 4). For example, the constraint:

   ```
   <IsInstanceOf>xyz:Harvest</IsInstanceOf>
   ```

   Is translated as the SPARQL triples:

   ```
   ?md a abc:MetadataRecord.
   ?ds abc:hasDescription ?md.
   ?ds a xyz:Harvest.
   ```

2. Alphanumeric attributes in the metadata record: The user can select one attribute in the metadata record, and perform a string matching, using the operators *PropertyIsEqualTo* and *PropertyIsLike* (See figure 5). In the later case the SPARQL implementation will require the definition of a *FILTER*.For instance the constraint:

   ```
   <PropertyIsLike>
   <ValueReference>dc:title</ValueReference>
   <Literal>water</Literal>
   </PropertyIsLike>
   ```

   would be translated to:

   ```
   ?md a abc:MetadataRecord.
   ?md dc:title ?xTitle.
   FILTER(regex(?xTitle,"water","i"))
   ```

3. Using toponym elements and spatial relationships: In this case the user identifies a toponym of interest, then she defines a spatial relationship between the bounding box of the metadata record and the geometry of the toponym of interest. We implement the spatial operators *Disjoint*,*Intersects*, *Contains* and *Within* (See figure 6). For example, we might be interested in metadata records whose bounding box intersect the geometry of *Australia*:

   ```
   <sfIntersects>
   <ValueReference>BoundingBox</ValueReference>
   <ToponymUnit>Australia</ToponymUnit>
   </sfIntersects>
   ```

   The constraint will be translated as GeoSPARQL:

   ```
   ?md a abc:MetadataRecord.
   ?md geo:hasGeometry ?boundingbox.
   ?boundingbox geo:asWKT ?boundingbox_wkt.
   ?topoUnit a tu:ToponymUnit.
   ?topoUnit tu:CountryName "Australia".
   ?topoUnit geo:hasGeometry ?topoGeo.
   ?topoGeo geo:asWKT ?topoWKT.
   FILTER(geof:sfIntersects
   (?boundingbox_wkt,?topoWKT))
   ```

Multiple constraints can be linked using the operators *AND* and *OR*.. When the constraint is completed the user submits it as a *POST*. The following XML code represents a query as formatted by the JavaScript running on the website.

```
<GetRecords>
<Query>
<Constraint><Filter><And>
<PropertyIsLike>
<ValueReference>dc:title</ValueReference>
<Literal>water</Literal>
</PropertyIsLike>
<IsInstanceOf>xyz:Harvest</IsInstanceOf>
<sfIntersects>
<ValueReference>BoundingBox</ValueReference>
<ToponymUnit>Australia</ToponymUnit>
</sfIntersects>
</And></Filter></Constraint>
</Query>
</GetRecords>
```

The translation of the constraint as SPARQL/GeoSPARQL is:

```
SELECT DISTINCT
?md ?xTitle
Where
{?md a abc:MetadataRecord.
?md dc:title ?xTitle.
?md geo:hasGeometry ?boundingbox.
?boundingbox geo:asWKT ?boundingbox_wkt.
?ds abc:hasDescription ?md.
?ds a xyz:Harvest.
?topoUnit a tu:ToponymUnit.
?topoUnit tu:CountryName "Australia".
?topoUnit geo:hasGeometry ?topoGeo.
?topoGeo geo:asWKT ?topoWKT.
FILTER((regex(?xTitle,"water","i"))&&
(geof:sfIntersects(?boundingbox_wkt,?topoWKT)))}
```

The response from the servlet is visualized in the website allowing the user to examine the metadata records (See figure 7).

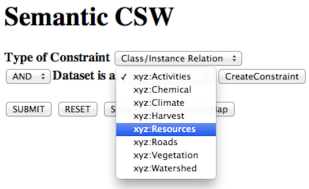

Figure 4: HTML user interface: Defining a constraint using the class membership.

Figure 5: HTML user interface: Defining a constraint using Dublin Core element.

3.5 Smart Queries

A *smart query* requires the combination of diverse datasources. However, first the researcher must be able to identify the most suitable dataset for the analysis. Our implementation aims to help users in this task. By using a domain ontology we improve the user's query capabilities. Our use of toponyms, allows the user to select areas of interest by name, and establish specific spatial relationships with the dataset of interest. The actual features of the dataset can later be obtained using the value of *abc:GetFeaturesURL* , a metadata record component.

4 CONCLUSIONS

In this work we present a simplified CSW implementation with a triplestore as a metadata repository. Our implementation has a working translator that is able to convert Filter Encoding queries into SPARQL/GeoSPARQL ones. The system allows

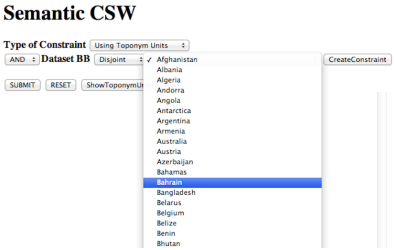

Figure 6: HTML user interface: Defining a constraint using a spatial relationship with an element of known toponym.

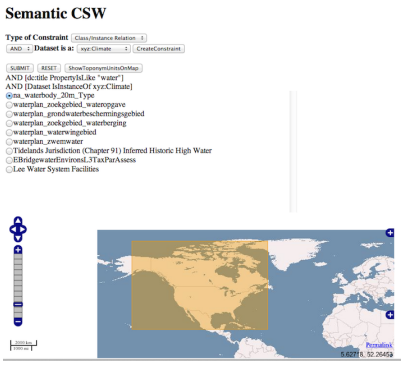

Figure 7: HTML user interface

complex queries that can take advantage of inference mechanisms provided by Semantic Web technologies.

At this point, our system only uses inference based on class to subclass relationships. However, we plan to extend these capabilities to include relationships between concepts, and automatic class membership determination.

Our approach to capture metadata information is generic, takes advantage of the OGC standard interfaces. With our harvesting tool we were able to create 2690 metadata records. However the information supplied by the WFS publishing entities has limitations and is in many cases incomplete. Our metadata records contain 1384 distinct keywords including 383 actual URLs. However in no case the URLs referred to any ontology or restricted formalized vocabulary. From the URLs, 217 were links to html documents, and 52 to XML documents. In both cases the documents contained extended metadata descriptions of the datasets. All the datasets with URL of extended descriptions were provided by one single WFS (giswebservices.massgis.state.ma.us/geoserver/wfs?), the rest of the keywords were strings with no formal semantics associated. Our metadata harvesting tool also obtained information regarding the names of the attributes of the dataset. In total we have obtained 6331 individual attribute names, all of them were strings with no formal semantics associated.

The use of extended descriptions in XML and HTML documents is not a standard practice among the WFS publishing entities. However, in case we find more documents of this kind, we can upgrade the harvesting tool in order to allow it to get information from the associated documents.

The results of our current implementation are promising, in the near future we will use a real domain class ontology and implement an automatic membership assignment based on harvested metadata.

ACKNOWLEDGEMENTS

This research is supported by: 1) Conseil régional de Bourgogne. 2) Direction Générale de l'Armement, see: http://www.defense.gouv.fr/dga/.

REFERENCES

DuCharme, B. (2011). *Learning SPARQL*. O'Reilly Media, Inc.

Dunne, D., Leadbetter, A., and Lassoued, Y. (2012). ICAN semantic interoperability cookbooks. Technical report, International Coastal Atlas Network.

ESRI (2010). GIS Best Practices: Spatial Data Infrastructure (SDI). http://www.esri.com/library/bestpractices/spatial-data-infrastructure.pdf. Accessed: July 2013.

ESRI, D. (2011). World administrative units. http://resources.arcgis.com/content/data-maps/10.0/world. Accessed on May 2013.

Goodwin, J. (2005). What have ontologies ever done for us - potential applications at a national mapping agency. In *OWL: Experiences and Directions (OWLED)*.

Gwenzi, J. (2010). Enhancing spatial web seach with semantic web technology and metadata visualization.

Harbelot, B., Arenas, H., and Cruz, C. (2013). Semantics for spatio-temporal "smart queries". In *Proceedings of the 9th. International Conference on Web Information Systems and Technologies*.

Janowicz, K., Schade, S., Broring, A., Kebler, C., Maue, P., and Stasch, C. (2010). Semantic enablement for spatial data infrastructures. *Transactions in GIS*, 14(2):111–129.

Janowicz, K., Scheider, S., Pehel, T., and Hart, G. (2012). Geospatial semantics and linked spatiotemporal data - past, present and future. *Semantic Web - Interoperability, Usability and Applicability*, 3(4):1–10.

Kammersell, W. and Dean, M. (2007). Conceptual search: Incorporating geospatial data into semantic queries. In Scharl, A. and Tochtermann, K., editors, *The Geospatial Web*, Advanced Information and Knowledge Processing, pages 47–54. Springer London. 10.1007/978-1-84628-827-2_5.

Kolas, D. and Batle, R. (2012). GeoSPARQL user guide. http://ontolog.cim3.net/file/work/SOCoP/Educational/GeoSPARQL User Guide.docx Accessed on May 2013.

Kolas, D., Hebeler, J., and Dean, M. (2005). Geospatial semantic web: Architecture of ontologies. pages 183–194.

Lopez-Pellicer, F. J., Florczyk, A., Renteria-Aguaviva, W., Nogueras-Iso, J., and Muro-Medrano, P. R. (2010). CSW2LD: a Linked Data frontend for CSW.

OGC (2012). OGC Institutional Web Site. http://www.opengeospatial.org/. Accessed: September 2013.

OSGeo (2012). CQL. http://docs.geotools.org/latest/userguide/library/cql/cql.html. Accessed on November 2012.

Pigot, S. (2012). Using rdf as metadata storage. http://trac.osgeo.org/geonetwork/wiki/rdfstore. Accessed on May 2013.

Vretanos, P. A. (2005). Filter encoding implementation specification. online. Accessed on May 2013.

Werner, D., Cruz, C., and Nicolle, C. (2012). Ontology-based recommender system of economic articles. In *WEBIST 2012*, pages 725–728.

Yue, P., Di, L., Yang, W., Yu, G., and Zhao, P. (2006). Path planning for chaining geospatial web services. In *Proceedings of the 6th international conference on Web and Wireless Geographical Information Systems*, W2GIS'06, pages 214–226, Berlin, Heidelberg. Springer-Verlag.

Yue, P., Gong, J., Di, L., He, L., and Wei, Y. (2011). Integrating semantic web technologies and geospatial catalog services for geospatial information discovery and processing in cyberinfrastructure. *GeoInformatica*, 15:273–303. 10.1007/s10707-009-0096-1.

APPENDIX

List of WFS used in this research:

- `http://geocarto.igac.gov.co:8082/geoservicios/quinientos_mil/wfs?`
- `http://geoservices.knmi.nl/cgi-bin/SCIA__CONS_V___IMAP____L2__2004.cgi?`
- `http://geoservices.knmi.nl/cgi-bin/SCIA_L2_TDTNO2_2007.cgi?`
- `http://sig.gov.ar/geoserver/ows?`
- `http://frameworkwfs.usgs.gov/framework/wfs/wfs.cgi?`
- `http://giswebservices.massgis.state.ma.us/geoserver/wfs?`
- `http://geoservices.provincie-utrecht.nl/arcgis/services/wfs_w01_water/MapServer/WFSServer?`
- `http://geocarto.igac.gov.co:8082/geoservicios/sigm/wfs?`
- `http://www.geoportaligm.gob.ec/regional/wfs?`
- `http://preview.grid.unep.ch:8080/geoserver/wfs?`
- `http://nsidc.org/cgi-bin/atlas_north?`
- `http://afromaison.grid.unep.ch/geoserver/ows?`
- `http://geowww.agrocampus-ouest.fr/geoserver/ows?`
- `http://www.sandre.eaufrance.fr/sdiger?`
- `http://www.fao.org/figis/geoserver/ows?`
- `http://ecos.fws.gov/geoserver/ows?`