Open Geospatial Consortium

Publication Date: 2015-11-18

Approval Date: 2015-09-17

Posted Date: 2015-07-17

Reference number of this document: OGC 15-057

Reference URL for this document: www.opengeospatial.net/doc/PER/t11-social-media-in-EM

Category: Public Engineering Report

Editors: Matthes Rieke;Simon Jirka;Stephane Fellah

OGC® Testbed-11 Incorporating Social Media in Emergency Response Engineering Report

Document type:	OGC® Engineering Report
Document subtype:	NA
Document stage:	Approved for public release
Document language:	English

Preface

This OGC Engineering Report (ER) was created as a deliverable for the OGC Testbed 11 initiative of the OGC Interoperability Program. This ER describes an approach for incorporating Social Media for Emergency Response applications that use spatial data infrastructures. This document also reports on findings about the advancements using Social Media and VGI resources. The ER includes ideas on improving the architecture, service change recommendations (primarily concerning the OGC Sensor Observation Service (SOS) 2.0 interface), and lessons learned.

This is not a normative document.

Suggested additions, changes, and comments on this Engineering Report are welcome and encouraged. Such suggestions may be submitted by email to the OGC.

License Agreement

Contents

Page

Figures

Page

Tables

Page

Listings

OGC® Testbed-11 Incorporating Social Media in Emergency Response Engineering Report

1 Introduction

1.1 Abstract

This OGC® Engineering Report (ER) describes approaches to incorporating Social Media as a data source for Emergency Response. The work described in the ER was performed as part of the Testbed-11 Cross-Community Interoperability service architecture. This ER provides an overview of the data sources for Social Media, discussing discovery aspects as well as quality assessment approaches.

Two approaches were implemented during Testbed-11. The first was based on the concept of using an OGC Sensor Observation Service (SOS) for handling *humans as sensors* and considered especially from an interoperability perspective. A mapping into the OGC/ISO Observations and Measurements (O&M) data model is presented and illustrated with examples from different social media platforms. This is complemented by a description of how data loading from social media platforms into an SOS server can be achieved. The second approach is based on using Linked Data to integrate content as social "objects" produced by the different social media sites. The ontology for describing social objects and activities is based on the SocialML ontology[1] and can be extended to accommodate new activities and social objects. The integration of social site content was accomplished with RDF scrapers accessible through a REST API. The merged knowledgebase can be accessed through a GeoSPARQL endpoint.

Another important topic of this report is the use of the OGC Web Processing Service (WPS) for analyzing the available social media data (i.e. detect clusters). Finally, a client for accessing the provided data sources and processes is introduced.

1.2 Business Value

The findings described in this ER will help to integrate social media content into standards-based geospatial information infrastructures and systems. This way, a new source of, often up-to-date, information will be made available. Especially in application contexts such as emergency response this will open up new opportunities to achieve better situational awareness. As a result, not only the capabilities of emergency responders will be improved but also the value of social media content will grow.

[1] Released by ImageMatters in the context of Testbed-11 (http://www.imagemattersllc.com/index.php/geospatial-semantics/)[2] https://wiki.52north.org/bin/view/SensorWeb/SensorObservationServiceIVDocumentation

1.3 Keywords

Ogcdoc, ogc documents, testbed 11, t11, ogc, cci, social media

1.4 Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Matthes Rieke (MRI)	52°North, m.rieke<at>52north.org
Simon Jirka (SJI)	52°North, s.jirka<at>52north.org
振宇 How	GIS.FCU, how<at>gis.tw
Stephane Fellah (STF)	Image Matters, stephanef<at>imagemattersllc.com

1.5 Future work

For recommendations on future work please refer to section 8.

1.6 Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

2 References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

OGC 06-121r3, OWS Common Implementation Standard

OGC 10-004r3, OGC Abstract Specification, Geographic information - Observations and Measurements

OGC 10-025r1, Observations and Measurements - XML Implementation

OGC 12-006, OGC Sensor Observation Service Interface Standard

OGC 14-016, OGC Testbed-10 CCI VGI Engineering Report

OGC 14-065, OGC WPS 2.0 Interface Standard

SIOC, http://www.w3.org/Submission/sioc-spec/

W3C Resource Description Framework (RDF), http://www.w3.org/RDF/

W3C Web Ontology Language (OWL), http://www.w3.org/2001/sw/wiki/OWL

3 Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r3] shall apply. In addition, the following terms and definitions apply.

4 Conventions

4.1 Abbreviated terms

API	Application Programming Interface
COI	Community of Interests
CRUD	Create, Read, Update and Delete
FOAF	Friend of a Friend
JSON	JavaScript Object Notation
JSON-LD	JSON for Linking Data
LDP	Linked Data Platform
MOC	Minimal Ontological Commitment
O&M	Observations & Measurements
OGC	Open Geospatial Consortium
OWL	Web Ontology Language
RDF	Resource Description Framework
SIOC	Semantically-Interlinked Online Communities
SKOS	Simple Knowledge Organization System
SNA	Social Network Analysis
SOS	Sensor Observation Service
Testbed-11	OGC Testbed, Phase 11
URL	Uniform Resource Locator
VGI	Volunteered Geographic Information
W3C	World Wide Web Consortium
WFS	Web Feature Service
WMS	Web Map Service

WPS	Web Processing Service
XML	Extensible Markup Language

4.2 UML notation

Some diagrams that appear in this document are presented using the Unified Modeling Language (UML) static structure diagram, as described in Subclause 5.2 of [OGC 06-121r3]. Additionally, UML sequence diagrams are used to illustrate workflow.

5 Testbed-11 Incorporating Social Media in Emergency Response Engineering Report

One task in the Testbed-11 CCI Thread was to study different approaches to incorporating social media information into Emergency Response applications. Two approaches were tested during the Testbed-11. The first one was based on the concept of using an OGC Sensor Observation Service (SOS) for handling *humans as sensors* is illustrated from an interoperability perspective. The developed O&M-based data model is presented and illustrated with examples from different social media platforms. This is complemented by a description of how data loading from social media platforms into an SOS server can be achieved. The second approach was based on using Linked Data to integrate social objects produced by the different social media networks. The ontology for describing social objects and activities is based on SocialML ontologies that can be extended to accommodate new activities and social objects. The integration of social sites was done through the use of RDF scrapers accessible through a REST API. The resulting knowledgebase was made available through a GeoSPARQL endpoint. Another important topic of this report is the use of the OGC Web Processing Service (WPS) for analyzing the available social media data (i.e. detect clusters). Finally, a client for accessing the provided data sources and processes is introduced.

5.1 CCI Architecture

The Testbed-11 Cross Community Interoperability (CCI) thread is organized into several different tracks following separate approaches that use various technology stacks. The different tracks are:

- Linked Data & Semantic Enablement of OGC Services
- Symbology & Semantic Mediation
- Aviation
- Use of Semantic Linked Data with RDF for National Map NHD and Gazetteer Data
- Social Media & Semantic Mediation

For information on the CCI tracks not covered in this document, consider the following Testbed-11 Engineering Reports:

- **OGC 15-054**: Implementing Linked Data and Semantically Enabling OGC Services
- **OGC 15-055**: Catalogue Service Analysis and Recommendations
- **OGC 15-066**: Use of Semantic Linked Data with RDF for National Map NHD and Gazetteer Data
- **OGC 15-058**: Symbology Mediation Engineering Report

The Social Media track features an emergency response use case: the evacuation of populations during a flood event (using historic data from San Francisco, December 11th, 2014). Figure 1 illustrates the overall architecture. The architecture involves client, server and processing/middleware components. Section 5 introduces the developed components in more detail.

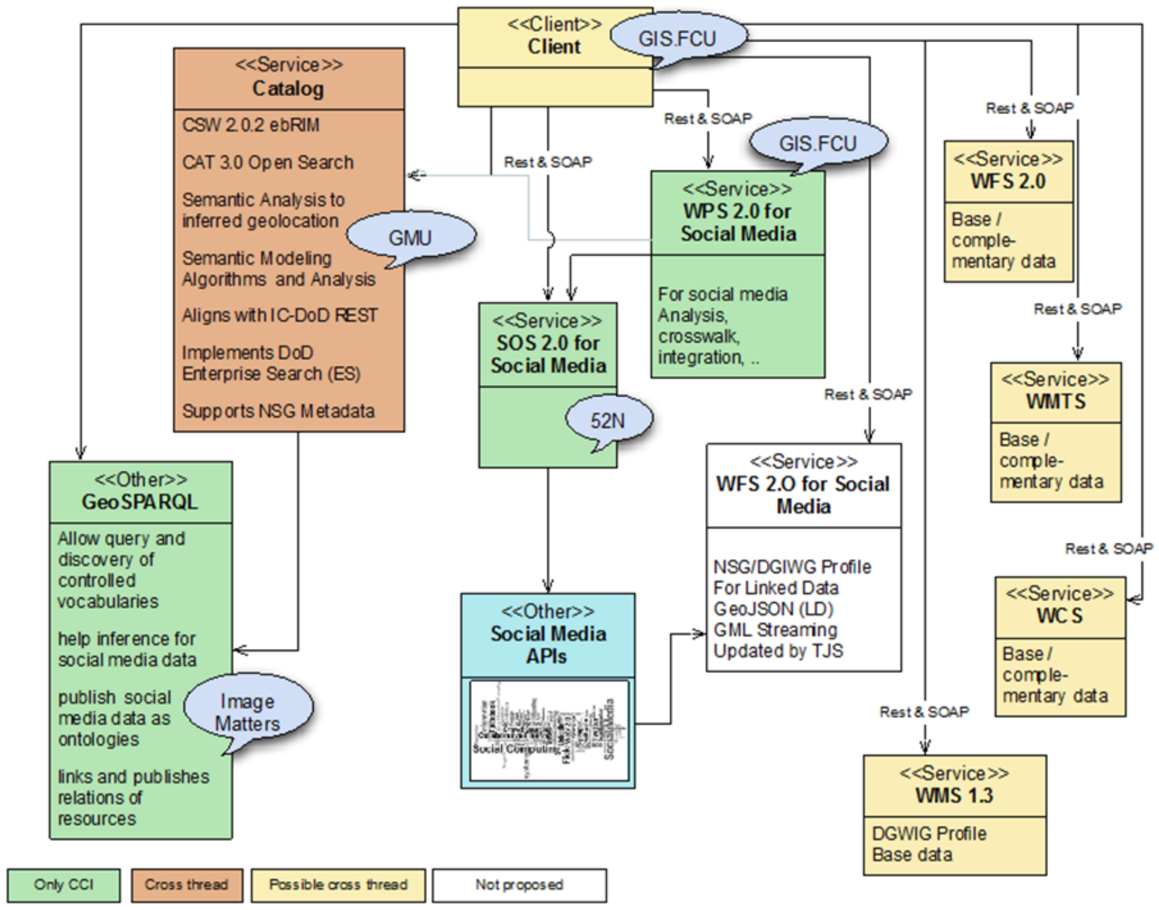

<<Client>>
Client
GIS.FCU
<<Service>>
Catalog
CSW 2.0.2 ebRIM
CAT 3.0 Open Search
Semantic Analysis to inferred geolocation
GMU
Semantic Modeling Algorithms and Analysis
Aligns with IC-DoD REST
Implements DoD Enterprise Search (ES)
Supports NSG Metadata
Rest & SOAP
<<Service>>
WPS 2.0 for Social Media
For social media Analysis, crosswalk, integration, ..
<<Service>>
SOS 2.0 for Social Media
52N
<<Service>>
WFS 2.0
Base / comple-mentary data
<<Service>>
WMTS
Base / comple-mentary data
<<Service>>
WCS
Base / comple-mentary data
<<Service>>
WFS 2.0 for Social Media
NSG/DGIWG Profile For Linked Data GeoJSON (LD) GML Streaming Updated by TJS
<<Service>>
WMS 1.3
DGWIG Profile Base data
<<Other>>
GeoSPARQL
Allow query and discovery of controlled vocabularies
help inference for social media data
publish social media data as ontologies
links and publishes relations of resources
Image Matters
<<Other>>
Social Media APIs
Only CCI
Cross thread
Possible cross thread
Not proposed

Figure 1 - CCI Architecture

5.2 Social Media Resources

The data to support the use case was harvested from well-established Social Media platforms. In order to identify valuable platforms that fit into the use case (e.g. geo-tagged data, high usage rate) candidate platforms were analyzed and assessed for their ability to best serve the use case.

This section provides an overview on the Social Media platforms that have been evaluated against the requirements for the Emergency Response scenario. The evaluation is separated into three categories – supported, promising and unsuitable platforms.

A decision sheet is documented for every platform, providing formal parameters for assessing the suitability.

5.2.1 Search Criteria

To support the use case of identifying possible evacuation routes for a flood event, a pre-defined search on all supported platforms was used to gather the data. The search criteria included a time window (2014-12-10 until 2014-12-12) and a geographic area (San Francisco center with radius of five miles around the center). No content-wise filtering (e.g. tags or textual content) was applied in the data harvesting process.

5.2.2 Supported Platforms

The platforms presented within this section are assessed as well-suited for Emergency Response applications. Besides the good suitability, open source implementations on gathering its data have been developed within the scope of Testbed-11.

The following criteria were used to identify suitable candidates of data for use in Emergency Response:

- Public Search API available
- Support for filter by geolocation
- Support for filter by time and date
- Support for filter by keywords/tags

Besides these hard criteria, additional soft criteria have been applied:

- Amount of overall postings for the desired time and location
- Number of postings with geotags
- Provision of pictures

5.2.2.1 Instagram

Instagram focuses on sharing photos and videos. Users can subsequently interact with these shared resources. Instagram has a growing user base and the current number of registered accounts is around 300 million.

Instagram was assessed to be a suitable candidate due to its features:

Table 1 - Assessment of Instagram

Criterion	**Availability**	**Comment**
Public search API	Available	
Filter by geolocation	Available	
Filter by time and date	Available	
Filter by keywords/tags	Available	
Number of postings	Ca. 150 for the defined search criteria	
Number of postings with geo information	Not measurable, no official information available	
Pictures supported	Yes	

5.2.2.2 Flickr

Similar to Twitter the usage of Flickr as a data source for Volunteered Geographic Information (VGI) was evaluated and tested in OGC Testbed-10 (see OGC 14-016). Flickr is the biggest photo community on the web and also provides asocial media features.

Flickr was assessed as a suitable candidate due to its features:

Table 2 - Assessment of Flickr

Criterion	**Availability**	**Comment**
Public search API	Available	
Filter by geolocation	Available	
Filter by time and date	Available	
Filter by keywords/tags	Available	
Amount of postings	Ca. 600 for the defined search criteria	
Number of postings with geo information	Not measurable, no official information available	
Pictures supported	Yes	

5.2.2.3 Twitter

Twitter is currently one of the largest Social Media networks with more than 300 million active users. The integration of Tweets into an OGC service architecture was prototyped within the OGC Testbed-10.

Twitter was assessed as a suitable candidate due to its features:

Table 3 - Assessment of Twitter

Criterion	Availability	Comment
Public search API	Available	
Filter by geolocation	Available	
Filter by time and date	Available	The REST API only supports a time frame for the last 7 days. Still the website provides earlier matches as well.
Filter by keywords/tags	Available	
Amount of postings	More than 3000 for the defined search criteria	
Number of postings with geo information	1% in 2013 (Hawelka et al. 2014), constantly growing (Thom et al. 2015)	Twitter resolves location from coordinates or metadata. For this testbed, Image Matters also geocoded Tweets without location information based on the hometown information associated with user accounts
Pictures supported	Yes	

5.2.3 Promising Platforms

Some platforms have been identified as good candidates for a source of data for integration into the service architecture. However an implementation has not been conducted.

5.2.3.1 Picasa

Picasa was assessed as a promising candidate due to its features:

Table 4 - Assessment of Picasa

Criterion	Availability	Comment
Public search API	Available	
Filter by geolocation	Available	
Filter by time and date	Available	
Filter by keywords/tags	Available	
Amount of postings	n/a	
Number of postings with geo information	Not measurable, no official information available	
Pictures supported	yes	

5.2.3.2 Open Street Map (OSM)

Open Street Map is a special candidate among the assessed platforms. OSM does not offer a public API comparable the other Social Media platforms. Still it provides up-to-date infrastructure data based on volunteered geographic information. Therefore OSM is a good candidate for inclusion into the overall scenario but should be treated differently (e.g. as a basis for identifying escape routes).

5.2.3.3 Storyful

Storyful was assessed as a promising candidate due to its features:

Table 5 - Assessment of Storyful

Criterion	Availability	Comment
Public search API	Available	
Filter by geolocation	Available	
Filter by time and date	Available	
Filter by keywords/tags	Available	
Amount of postings	n/a	
Number of postings with geo information	Not measurable, no official information available	
Pictures supported	yes	

5.2.3.4 Panoramio

Panoramio was assessed as a promising candidate due to its features:

Table 6 - Assessment of Panoramio

Criterion	Availability	Comment
Public search API	Available	
Filter by geolocation	Available	
Filter by time and date	Available	
Filter by keywords/tags	Available	
Amount of postings	n/a	
Number of postings with geo information	Not measurable, no official information available	
Pictures supported	yes	

5.2.3.5 YouTube

Though the number of YouTube videos featuring geolocation and near real time footage is increasing, the data supporting the defined scenario was very small. The integration was tested and performed prototypically within the Linked Data components described in section 5.7. In the future and also for other scenarios the amount of relevant data might grow. Thus the platform was assessed as promising.

Table 7 - Assessment of Yotube

Criterion	Availability	Comment
Public search API	Available	
Filter by geolocation	Available	
Filter by time and date	Available	
Filter by keywords/tags	Available	
Amount of postings	n/a	
Number of postings with geo information	Not measurable, no official information available	
Pictures supported	yes (videos)	

5.2.4 Unsuitable Platforms

The following platforms did not meet the requirements for providing public data useful for Emergency Response scenarios.

5.2.4.1 Facebook

Though Facebook is the biggest Social Media network in the Web, it does not meet the requirements for an Emergency Response scenario as it closed its public search API in April 2014.

5.2.4.2 Tumblr

Tumblr is a personal blogging system with community aspects. Still it does not meet the specified requirements as no global search API is available. A search is limited to a single user blog instance only.

5.2.4.3 Pinterest

Similar to Tumblr, Pinterest does not provide a global search but only a limited discovery to specific user domains.

5.2.4.4 Snapchat

Snapchat cannot be considered a classic Social Media platform as there are no persisting postings available. It is designed in a way that media only exists for a predefined amount of time. Snapchat also does not feature any kind of API.

5.3 SOS for Social Media

The first approach investigated for incorporating social media with emergency response applications leveraged the existing Sensor Observation Service (SOS), by modeling social media information as Observations, extending the role of human as sensor.

The SOS for Social Media was set-up based on the 52°North SOS 4.x development line[2]. This SOS server is implemented in Java relying on several standard technologies and libraries. The architecture of the 52°North SOS server 4.x is summarized in Figure 2.

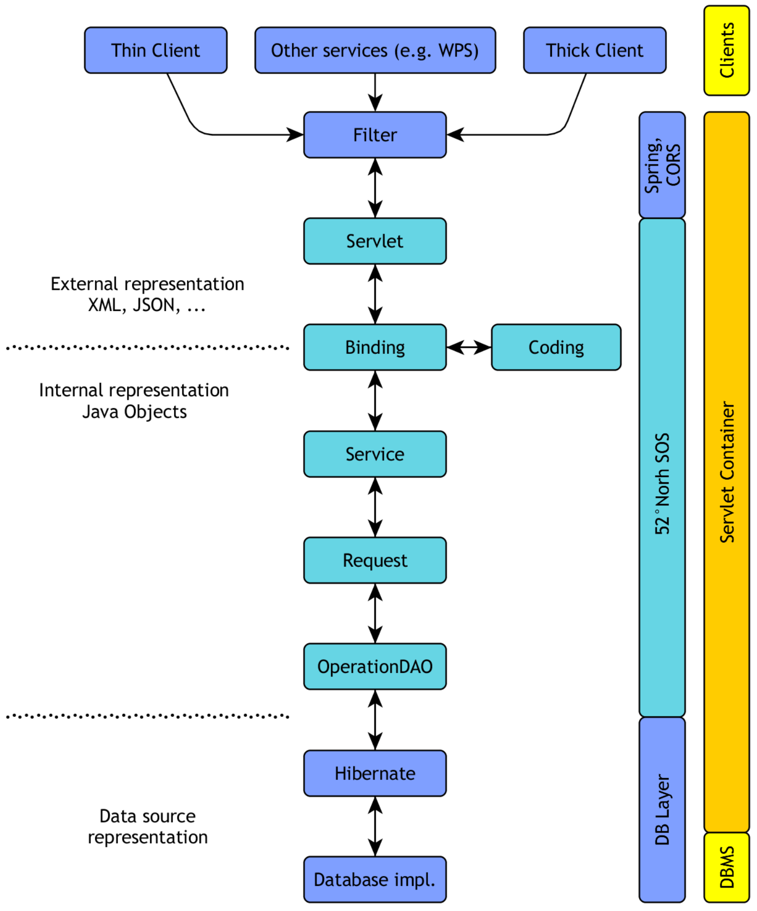


Figure 2 - Architecture of the 52°North SOS

The 52°North SOS server can be flexibly coupled to a broad range of data sources (e.g. many commonly used relational database management systems) and data models. For the OGC Testbed-11 the default database model of the 52°North SOS server was used within a PostgreSQL database. As this data model was designed based on the concepts provided by the O&M 2.0 standard, no further changes to the internal data model of the 52°North

2 https://wiki.52north.org/bin/view/SensorWeb/SensorObservationServiceIVDocumentation

SOS were necessary. This approach relies on the findings described in the OGC® Testbed-10 CCI VGI Engineering Report.

The SOS for Social Media supports the following operations specified by the SOS 2.0 standard:

- GetCapabilities: Retrieving metadata about the SOS server
- DescribeSensor: Retrieving metadata about the observers
- GetObservation: Retrieving social media observations/content

In addition, the SOS for Social Media offers the following transactional SOS operations to publish previously harvested social media observations:

- InsertSensor: Registration of new observers
- InsertObservation: Inserting new social media content published by a previously registered observer

Furthermore, to facilitate the integration with legacy software that does not yet support the SOS 2.0 standard, the WFS 2.0 GetFeature operation was also implemented.

Thus, the SOS for social media allows interoperable publication and retrieval of social media observations. As the next subsection will show, the O&M 2.0 data model is already well suited for modelling and encoding social media observations. No significant changes to the SOS implementation were needed to handle social media content. The main challenges were on the harvesting and data loading mechanisms described in section 5.4.

5.3.1 Mapping of Social Media entries to O&M 2.0

The mapping of social media content to the O&M 2.0 model can be achieved in a straightforward manner. This is based on the idea that basically all social media content relevant for the scenario (e.g. photos, messages describing certain situations) can be considered as observations made by humans. This leads to the following mapping between typical social media content (see Figure 3 for an example based on Flickr) and the O&M 2.0 model:

- Feature of Interest: The location/object/place for which the observer has published an observation
- Observed Property: The property which is described by the observation (e.g. in case of a photo this would be the visual perception)
- Procedure: The observer who made the observation (e.g. the photographer)
- Result: A reference to the social media content (e.g. a link to the photo on Flickr)

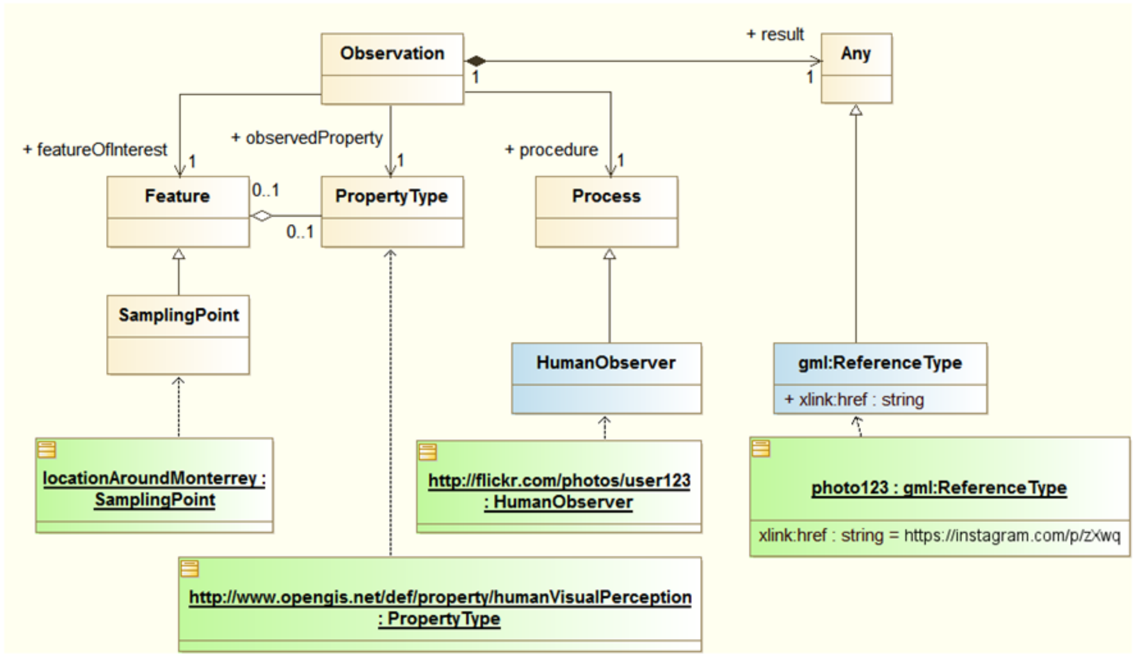

Observation
+ result
Any
1
1
+ featureOfInterest
+ observedProperty
+ procedure
Feature
0..1
0..1
PropertyType
Process
SamplingPoint
HumanObserver
gml:ReferenceType
+ xlink:href : string
locationAroundMonterrey : SamplingPoint
http://flickr.com/photos/user123 : HumanObserver
photo123 : gml:ReferenceType
xlink:href : string = https://instagram.com/p/zXwq
http://www.opengis.net/def/property/humanVisualPerception : PropertyType

Figure 3 - Example Flickr Observation

5.3.2 Examples

The two following subsections show an example for a GetObservation request as well as a response.

5.3.2.1 Example Request - GetObservation

The following URL shows an exemplary GetObservation request (KVP binding) to retrieve content from the SOS for Social Media:

```
http://ows.dev.52north.org:8080/52n-wfs-webapp/sos/kvp
?service=SOS
&version=2.0.0
&request=GetObservation
&observedProperty=http://www.opengis.net/def/property/
                  humanVisualPerception
&temporalFilter=om:phenomenonTime,2014-12-11T10:00:00.000Z/
                2014-12-11T15:00:00.000Z
```

In this case, all human observations made between 10:00 and 15:00 on the 11th December 2014 are requested.

5.3.2.2 Example Response - GetObservation

Listing 1 shows an excerpt from a SOS GetObservation response which contains one Flickr photograph. This observation has the following properties:

- The photo was taken on December 11th 2014 at 07:40:30. It was published at 21:17:43 on the same day.
- The photo was taken in San Francisco.
- The result points to the URL on Flickr where the photo can be accessed.

Listing 1 - Example Flickr Observation

```
<sos:observationData>
   <om:OM_Observation gml:id="o_346FA74773F61F3B190E724CBAC067F985C62">
      <gml:identifier codeSpace="http://www.opengis.net/def/nil/OGC/0/
                                 unknown">
         https://www.flickr.com/photos/username/123456789/
      </gml:identifier>
      <om:type xlink:href="http://www.opengis.net/def/observationType/
                           OGC-OM/2.0/OM_TextObservation"/>
      <om:phenomenonTime>
         <gml:TimeInstant gml:id="phenomenonTime_5515">
            <gml:timePosition>
               2014-12-11T07:40:30.000Z
            </gml:timePosition>
         </gml:TimeInstant>
      </om:phenomenonTime>
      <om:resultTime>
         <gml:TimeInstant gml:id="ti_46E6CA6A401C83E0612146D2D54CF02A">
            <gml:timePosition>
               2014-12-11T21:17:43.000Z
            </gml:timePosition>
         </gml:TimeInstant>
      </om:resultTime>
      <om:procedure xlink:href="https://www.flickr.com/photos/
                                123456789@N00"/>
      <om:observedProperty xlink:href="http://www.opengis.net/def/
                                       property/
                                       humanVisualPerception"/>
      <om:featureOfInterest>
         <sams:SF_SpatialSamplingFeature xmlns:sams="http://www.
                                                     opengis.net/
                                                     samplingSpatial/
                                                     2.0"
                                         xmlns:sf="http://www.opengis.
                                                   net/sampling/2.0"

                                         gml:id="ssf_B7227E403423A4AB">
            <gml:description>San Francisco</gml:description>
            <gml:identifier codeSpace="">
               7.MJR8tTVZu7O1EgB
            </gml:identifier>
            <gml:name>San Francisco</gml:name>
            <sf:type xlink:href="http://www.opengis.net/def/
                                 samplingFeatureType/
                                 OGC-OM/2.0/SF_SamplingPoint"/>
            <sf:sampledFeature xlink:href="http://52north.org/sos/
                                           featureOfInterest/world"/>
            <sams:shape>
               <gml:Point gml:id="point_ssf_B7227E403423A4AB6CFFCD676">
```

```
                  <gml:pos srsName="http://www.opengis.net/def/crs/
                                    EPSG/0/4326">
                     -122.3926 37.7927
                  </gml:pos>
               </gml:Point>
            </sams:shape>
         </sams:SF_SpatialSamplingFeature>
      </om:featureOfInterest>
      <om:result xmlns:xs="http://www.w3.org/2001/XMLSchema"
                 xsi:type="xs:string">
         https://www.flickr.com/photos/username/123456789/
      </om:result>
   </om:OM_Observation>
</sos:observationData>
```

5.4 Integrating Social Media Data using SOS

In order to smoothly integrate different social media platforms, 52°North developed a Java API for Social Media Harvesting[3]. This section illustrates the architecture, concepts and workflows of this API.

5.4.1 Class Architecture

Figure 4 shows the high level class structure of the API. The central component is the `HarvesterController`. It manages the available implementations of the `Harvester` interface. The `HarvesterController` can be configured with several search criteria (see Section 5.2.1) and provides one single `harvest` method for starting the overall harvesting process.

The `Harvester` interface is the entrance point for the access to the different Social Media APIs. It provides high level methods for dataset discovery (`searchForPostingssAt()`).

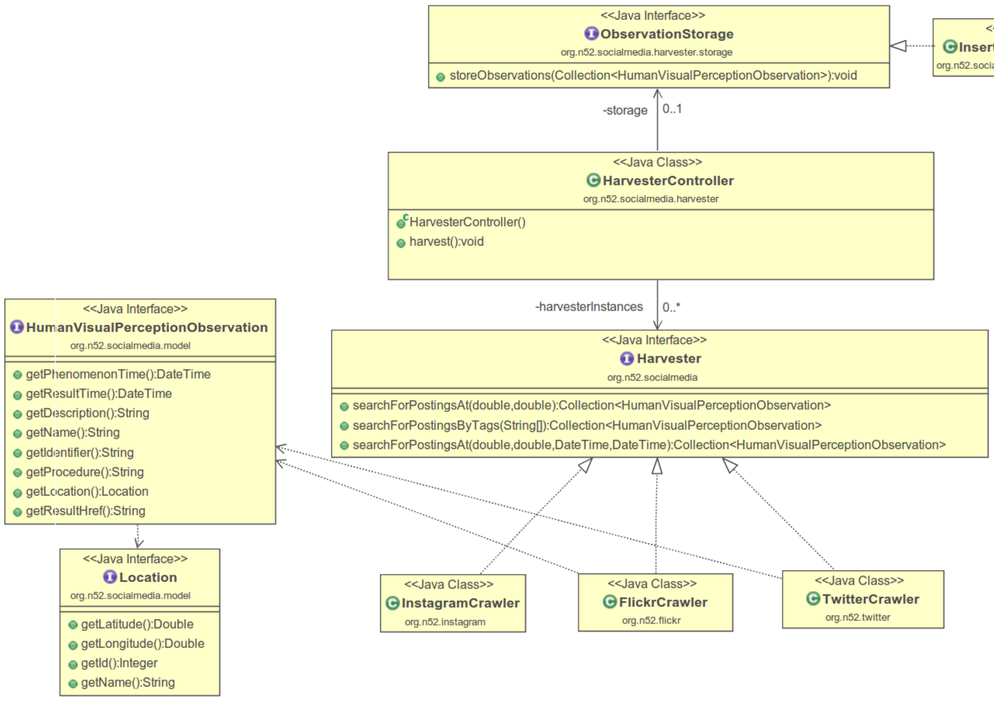


Figure 4 - Harvester API class diagram

All `Harvester` implementations shall provide a collection of `HumanVisualPerception-Observation` instances back to the `HarvesterController` after finishing the search. This interface provides abstract methods to data concepts that are shared among social media APIs (e.g. user, tags, location and time) as well as methods for accessing the actual content of a posting (e.g. image captions, post contents). The `HumanVisualPerception-`

[3] https://github.com/52North/ows-11-social-media-harvester

`Observation` interface is aligned with the O&M 2.0 Social Media model (see section 5.3.1) and provides corresponding method names.

After the harvest process has finished, the `HarvesterController` can upload the data to an SOS 2.0 server through the transactional interface via the `InsertObservationToSOS` class. Still, the `ObservationStorage` interface provides means to extend the architecture with a complementary storage mechanism.

5.4.2 API connectors

The developed API connectors all implement the aforementioned `Harvester` interface. All supported Social Media APIs have been accessed via a RESTful interfaces providing JSON encoded data. For accessing Twitter a dedicated Java library[4] was used.

Instagram and Flickr were accessed by straight-forward URL calls. An exemplary location-based search on Instagram looks like the following:

https://api.instagram.com/v1/media/search?lat=37.7611895&lng=-122.3821116&min_timestamp=1449705600&max_timestamp=1449878400&access_token=

The search resource provides a JSON similar to the following:

```
{
   "meta":{
      "code":200
   },
   "data":[
      {
         "attribution":null,
         "tags":[
            "rain",
            "storm",
            "flood"
         ],
         "type":"image",
         "location":{
            "latitude":37.7611895,
            "name":"San Francisco",
            "longitude":122.3821116,
            "id":1234
         },
         "comments":{
            "count":0,
            "data":[

            ]
         },
         "filter":"Normal",
         "created_time":"1423480138",
         "link":"http:\/\/instagram.com\/p\/anonymous\/",
```

[4] http://twitter4j.org/

```
         "likes":{
            "count":2,
            "data":[
               {
                  "username":"anonymous",
                  "profile_picture":"***.jpg",
                  "id":"1234",
                  "full_name":"Anony mous"
               }
            ]
         },
         "images":{
            "low_resolution":{
               "url":"***.jpg",
               "width":306,
               "height":306
            },
            "thumbnail":{
               "url":"***.jpg",
               "width":150,
               "height":150
            },
            "standard_resolution":{
               "url":"***.jpg",
               "width":640,
               "height":640
            }
         },
         "users_in_photo":[

         ],
         "caption":{
            "created_time":"1423480138",
            "text":"Storm in SF!!!",
            "from":{
               "username":"anonymous",
               "profile_picture":"***.jpg",
               "id":"1234",
               "full_name":"Anony mous"
            },
            "id":"1234"
         },
         "user_has_liked":false,
         "id":"1234",
         "user":{
            "username":"anonymous",
            "website":"",
            "profile_picture":"***.jpg",
            "full_name":"Anony mous",
            "bio":"",
            "id":"1234"
         }
      }
   ]
}
```

This data was parsed and for each entry contained in the `data` array, a `HumanVisualPerceptionObservation` instance has been created.

5.4.3 SOS 2.0 Transactional Interface

For inserting the gathered data into the SOS 2.0 for Social Media the transactional interface of the SOS 2.0 interface standard was used. This way the implementation of harvesting and the storage is connected via interoperable and well-established methods. This section provides a brief overview on the required interface interactions.

5.4.3.1 InsertSensor

Prior to the insertion of actual observations knowledge of the `procedure` to the SOS service needs to be provided. The user who posted the social media entry has been mapped to the procedure in the Social Media model (see section 5.3.1). The following request illustrates an exemplary `InsertSensor` request for providing these metadata.

Listing 2 - InsertSensor Example

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3.org/2003/05/soap-envelope
http://www.w3.org/2003/05/soap-envelope/soap-envelope.xsd">
   <env:Body>
      <swes:InsertSensor service="SOS" version="2.0.0"
                         xmlns:swes="http://www.opengis.net/swes/2.0"
                         xmlns:sos="http://www.opengis.net/sos/2.0"
                         xmlns:swe="http://www.opengis.net/swe/1.0.1"
                         xmlns:sml="http://www.opengis.net/sensorML
                            /1.0.1"
                         xmlns:gml="http://www.opengis.net/gml"
                         xmlns:xlink="http://www.w3.org/1999/xlink"
                         xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                            instance"
                         xsi:schemaLocation="http://www.opengis.net/
                            sos/2.0
                            http://schemas.opengis.net/sos/2.0/
                            sosInsertSensor.xsd
                            http://www.opengis.net/swes/2.0
                            http://schemas.opengis.net/swes/2.0/
                            swes.xsd">
         <swes:procedureDescriptionFormat>
            http://www.opengis.net/sensorML/1.0.1
         </swes:procedureDescriptionFormat>
         <swes:procedureDescription>
            <sml:SensorML version="1.0.1">
               <sml:member>
                  <sml:System>
                     <sml:identification>
                        <sml:IdentifierList>
                           <sml:identifier name="uniqueID">
                              <sml:Term definition="urn:ogc:def:
                                          identifier:OGC:1.0:uniqueID">
                                 <sml:value>${procedure}</sml:value>
                              </sml:Term>
                           </sml:identifier>
                           <sml:identifier name="longName">
                              <sml:Term definition="urn:ogc:def:
                                        identifier:OGC:1.0:longName">
                                 <sml:value>
                                    Procedure ${procedure}
                                 </sml:value>
                              </sml:Term>
                           </sml:identifier>
                           <sml:identifier name="shortName">
                              <sml:Term definition="urn:ogc:def:
                                        identifier:OGC:1.0:shortName">
                                 <sml:value>
                                    Procedure ${procedure}
                                 </sml:value>
                              </sml:Term>
                           </sml:identifier>
                        </sml:IdentifierList>
                     </sml:identification>
                     <sml:capabilities name="offerings">
                        <swe:SimpleDataRecord>
                           <swe:field name="Offering for ${offering}">
                              <swe:Text definition="urn:ogc:def:
                                        identifier:OGC:offeringID">
                                 <gml:name>
                                    Offering for ${offering}
                                 </gml:name>
                                 <swe:value>${offering}</swe:value>
                              </swe:Text>
                           </swe:field>
                        </swe:SimpleDataRecord>
                     </sml:capabilities>
                     <sml:capabilities name="featuresOfInterest">
                        <swe:SimpleDataRecord>
                           <swe:field name="featureOfInterestID">
                              <swe:Text>
                                 <swe:value>
                                    http://www.52north.org/sos/
                                    featureOfInterest/world/us/ca/
                                    san_francisco
                                 </swe:value>
                              </swe:Text>
                           </swe:field>
                        </swe:SimpleDataRecord>
                     </sml:capabilities>
                     <sml:inputs>
                        <sml:InputList>
                           <sml:input name="test_observable
                                            _property_9">
                              <swe:ObservableProperty
                                 definition="http://www.opengis.net/def
                                     /property/humanVisualPerception"/>
                           </sml:input>
                        </sml:InputList>
                     </sml:inputs>
                     <sml:outputs>
```

```
                              <sml:OutputList>
                                 <sml:output name="humanVisualPerception">
                                    <swe:Category
                                       definition="http://www.opengis.net/def
                                          /property/humanVisualPerception">
                                       <swe:codeSpace xlink:href=
                                                     "NOT_DEFINED"/>
                                    </swe:Category>
                                 </sml:output>
                              </sml:OutputList>
                           </sml:outputs>
                        </sml:System>
                     </sml:member>
                  </sml:SensorML>
               </swes:procedureDescription>
               <swes:observableProperty>
                  http://www.opengis.net/def/property/humanVisualPerception
               </swes:observableProperty>
               <swes:metadata>
                  <sos:SosInsertionMetadata>
                     <sos:observationType>
                        http://www.opengis.net/def/observationType/OGC-
                        OM/2.0/OM_CategoryObservation
                     </sos:observationType>
                     <sos:featureOfInterestType>
                        http://www.opengis.net/def/samplingFeatureType/OGC-
                        OM/2.0/SF_SamplingPoint
                     </sos:featureOfInterestType>
                  </sos:SosInsertionMetadata>
               </swes:metadata>
            </swes:InsertSensor>
         </env:Body>
      </env:Envelope>
```

In the case of Social Media the `offering` was defined as the overall Social Media platform (e.g. Twitter, Instagram or Flickr).

5.4.3.2 InsertObservation

Once the `offering` and `procedure` are registered at the SOS 2.0, the actual observations can be inserted. This has been achieved via the `InsertObservation` method. The following request provides an exemplar template for inserting an observation.

Listing 3 - InsertObservation Example

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

              xsi:schemaLocation="http://www.w3.org/2003/05/soap-
                                  envelope
                                  http://www.w3.org/2003/05/
                                  soap-envelope/
                                  soap-envelope.xsd">
```

```
<env:Body>
   <sos:InsertObservation service="SOS" version="2.0.0"
                          xmlns:sos="http://www.opengis.net/sos/2.0"
                          xmlns:swes="http://www.opengis.net/swes/
                                2.0"
                          xmlns:swe="http://www.opengis.net/swe/2.0"
                          xmlns:sml="http://www.opengis.net/
                               sensorML/ 1.0.1"
                          xmlns:gml="http://www.opengis.net/gml/3.2"
                          xmlns:xlink="http://www.w3.org/1999/xlink"
                          xmlns:om="http://www.opengis.net/om/2.0"
                          xmlns:sams="http://www.opengis.net/
                               samplingSpatial2.0"
                          xmlns:sf="http://www.opengis.net/sampling/
                              2.0"
                          xmlns:xs="http://www.w3.org/2001/
                              XMLSchema"
                          xsi:schemaLocation="http://www.opengis.
                              net/sos/2.0
                              http://schemas.opengis.net/sos/
                              2.0/sos.xsd
                              http://www.opengis.net/
                              samplingSpatial/2.0
                              http://schemas.opengis.net/
                              samplingSpatial/2.0/
                              spatialSamplingFeature.xsd">
      <sos:offering>${offering}</sos:offering>
      <sos:observation>
         <om:OM_Observation gml:id="photo-295155260-observation">
            <gml:description>${description}</gml:description>
            <om:type xlink:href="http://www.opengis.net/def/
                        observationType/OGC-OM/2.0/
                        OM_CategoryObservation"/>
            <om:phenomenonTime>
               <gml:TimeInstant gml:id="phenomenonTime">
                  <gml:timePosition>${phenTime}</gml:timePosition>
               </gml:TimeInstant>
            </om:phenomenonTime>
            <om:resultTime>
               <gml:TimeInstant gml:id="resultTime">
                  <gml:timePosition>${resultTime}</gml:timePosition>
               </gml:TimeInstant>
            </om:resultTime>
            <om:procedure xlink:href="${procedure}"/>
            <om:observedProperty xlink:href="http://www.opengis.net/
                                   def/property/
                                   humanVisualPerception"/>
            <om:featureOfInterest>
               <sams:SF_SpatialSamplingFeature gml:id=
                                          "ssf_instance">
                  <gml:description>${description}</gml:description>
                  <gml:identifier codeSpace="">${identifier}
                     </gml:identifier>
                  <gml:name>${name}</gml:name>
```

```
                        <sf:type xlink:href="http://www.opengis.net/def
                                            /samplingFeatureType/OGC-
                                            OM/2.0/SF_SamplingPoint"/>
                        <sf:sampledFeature xlink:href="http://www.52north.
                                            org/sos/featureOfInterest/
                                            world/us/ca/san_francisco"/>
                        <sams:shape>
                           <gml:Point gml:id="foiPointID">
                              <gml:pos
                              srsName="http://www.opengis.net/def/crs/
                                       EPSG/0/4326">${lonLat}
                              </gml:pos>
                           </gml:Point>
                        </sams:shape>
                     </sams:SF_SpatialSamplingFeature>
                  </om:featureOfInterest>
                  <om:result xsi:type="gml:ReferenceType"
                             xlink:href="${resultHref}"/>
               </om:OM_Observation>
            </sos:observation>
         </sos:InsertObservation>
      </env:Body>
   </env:Envelope>
```

The different properties (e.g. identifier, offering and description) directly map to the methods of the `HumanVisualPerceptionObservation` interface.

5.5 WPS for Social Media

GIS.FCU implemented a WPS 2.0 instance based on GeoServer[5]. GeoServer is a Java based GIS web server. Applications can publish geospatial data easily by the functions and GUI built in GeoServer. The latest version of GeoServer is 2.7.1 and it supports various OGC standards, including WFS, WCS, WMS, WMTS, TMS and WPS. An algorithm for spatially clustering Social Media contents has been implemented. The operations GetStatus, GetResult and Dismiss were implemented in the latest version of the GeoServer WPS that supports the WPS 2.0 standard as well as a RESTful binding.

5.5.1 Cluster Algorithm

The DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm was chosen for the spatial clustering of Social Media contents provided by the data sources (SOS and GeoSPARQL endpoint). DBSCAN groups together points that are close to each other (see Figure 5). The resulting clusters are provided as a result set of the WPS process.

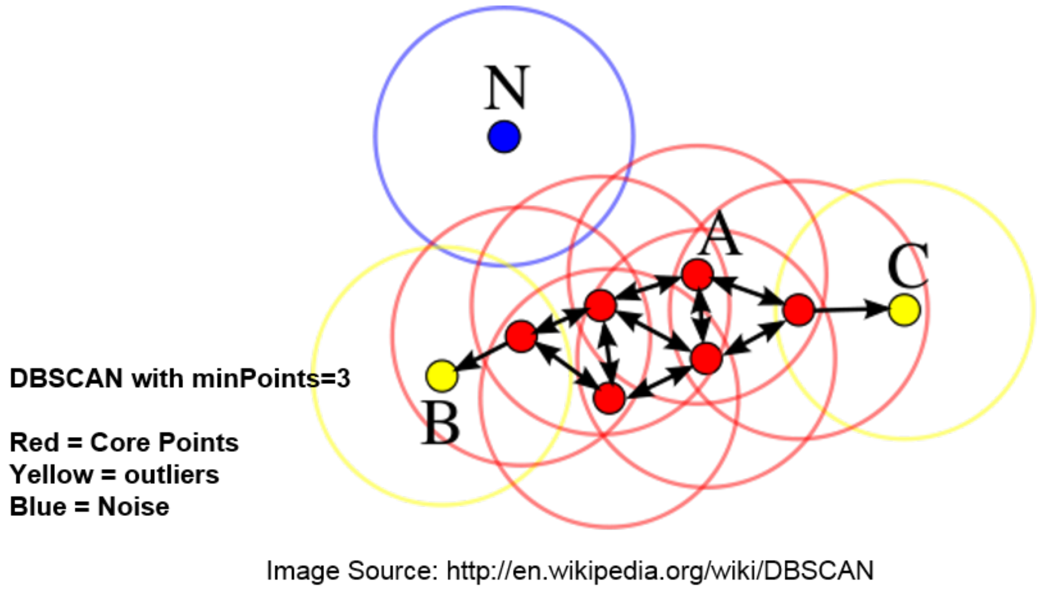


Figure 5 - DBSCAN illustration

5.5.2 Operations and Requests

Besides GetCapabilities and DescribeProcess, the Execute operation is the main entrance point for the algorithm. The identifier for the developed DBSCAN algorithm is “gs:SocialMedia”.

[5] http://geoserver.org/

“gs:SocialMedia” provides three parameters to calculate clustered social media data.

1. **Keyword**: filter parameter to extract data regarding a specific issue (e.g. “storm”)
2. **Distance**: the distance of radius for each social media data element to calculate the number of neighbors
3. **Neighbors**: the fulfilling condition if the number of social media data within specific radius distance exceeds neighbors

The Execute operation has three patterns based on different requirements:

- Get all social media data from SOS by sending request with null parameters.
- Get clustered social media data from SOS by sending request with distance, neighbors parameters.
- Get clustered social media data regards to specific issue from SOS by sending request with all parameters.

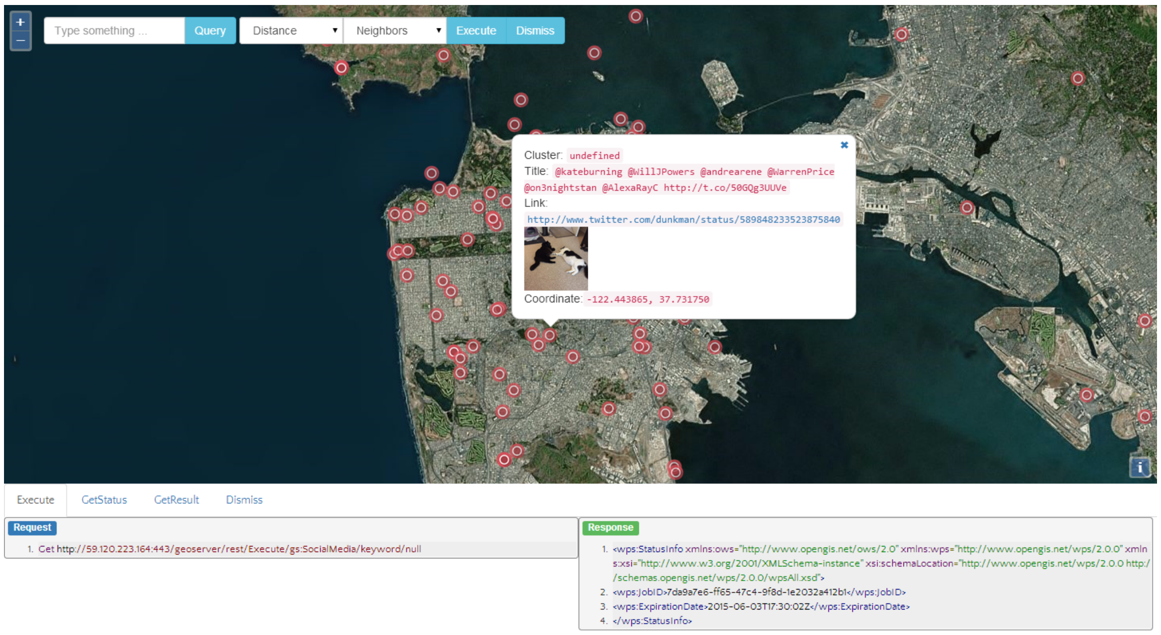
The result is retrieved asynchronously via the GetResult operation. An exemplary result is illustrated in Listing 4.

Listing 4 - Example of a WPS StatusInfo response

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:StatusInfo xmlns:ows="http://www.opengis.net/ows/2.0"
                xmlns:wps="http://www.opengis.net/wps/2.0.0"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://www.opengis.net/wps/2.0.0
                                    http://schemas.opengis.net/wps/
                                    2.0.0/wpsAll.xsd">
   <wps:JobID>082be5b5-44aa-4e22-8a70-0f779c651d2d</wps:JobID>
   <wps:Status>Succeeded</wps:Status>
   <cluster id="-1"/>
   <cluster id="0"/>
   <cluster id="1">
      <sensor id="1">
         <lat>37.771007</lat>
         <lng>-122.412694</lng>
         <title>Major Storm in San Francisco!!</title>
      </sensor>
   </cluster>
</wps:StatusInfo>
```

5.6 Social Media Client

The Social Media Client developed by GIS.FCU interacts with the 52°North SOS for Social Media and the Image Matters GeoSPARQL endpoint as well as the WPS for clustering. It supports the retrieval of all collected Social Media content (see Figure 6).



Cluster: undefined
Title: @kateburning @WillJPowers @andrearene @WarrenPrice @on3nightstan @AlexaRayC http://t.co/50GQg3UUVe
Link: http://www.twitter.com/dunkman/status/589848233523875840
Coordinate: -122.443865, 37.731750

```
1. Get http://59.120.223.164:443/geoserver/rest/Execute/gs:SocialMedia/keyword/null
```

```
1. <wps:StatusInfo xmlns:ows="http://www.opengis.net/ows/2.0" xmlns:wps="http://www.opengis.net/wps/2.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wps/2.0.0 http://schemas.opengis.net/wps/2.0.0/wpsAll.xsd">
2. <wps:JobID>7da9a7e6-ff65-47c4-9f8d-1e2032a412b1</wps:JobID>
3. <wps:ExpirationDate>2015-06-03T17:30:02Z</wps:ExpirationDate>
4. </wps:StatusInfo>
```

Figure 6 - Display of all Social Media content

The execution of the WPS process is done by providing the required parameters via the user interface on top of the map. Figure 7 shows the clustered results of the WPS process.

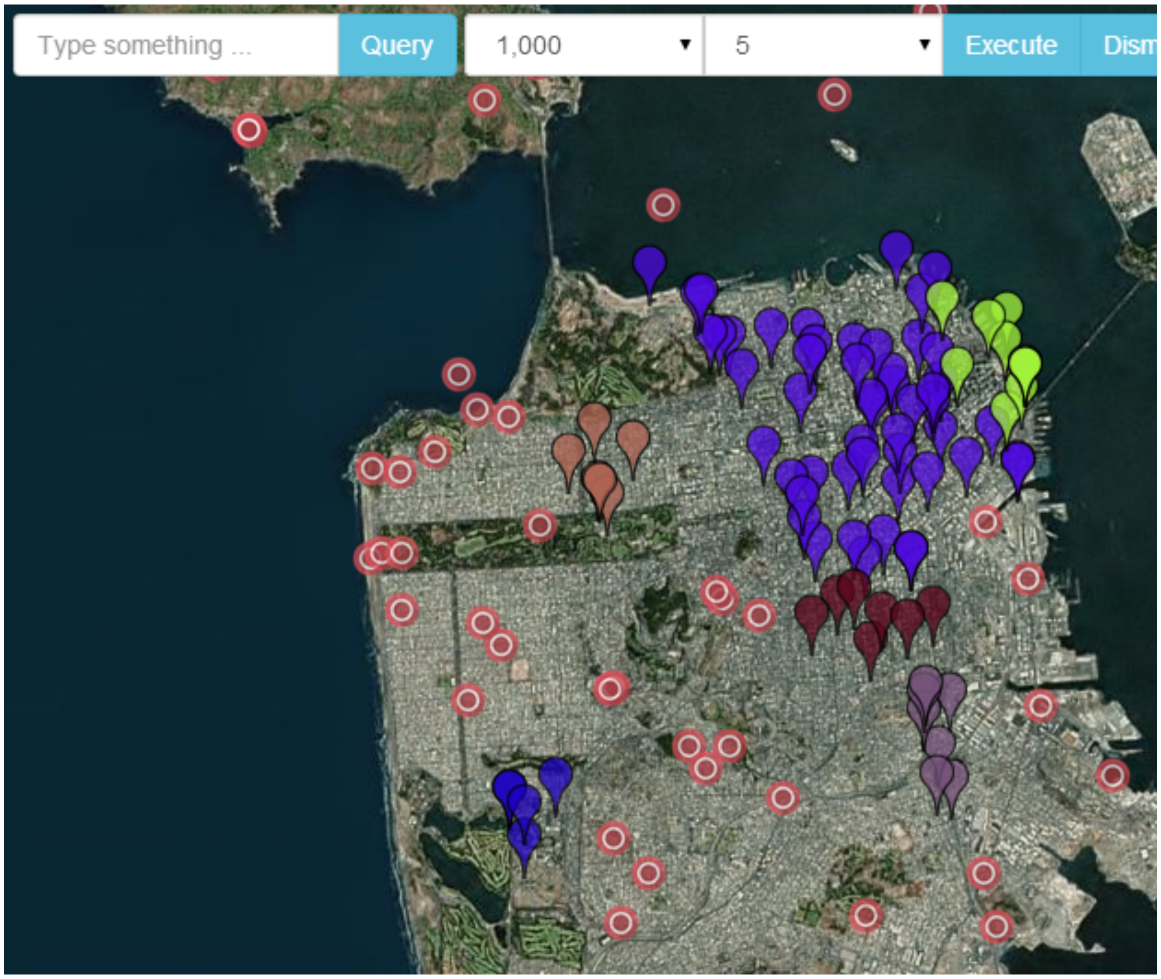


Figure 7 - Result of the DBSCAN Clustering

5.6.1 Workflow

Figure 8 illustrates the workflow and interactions with other components. The WPS process retrieves the relevant observations from the SOS 2.0 for Social Media and applies the clustering subsequently. While the calculation is ongoing, the client can retrieve the status of the process job as well as the result once it has finished.

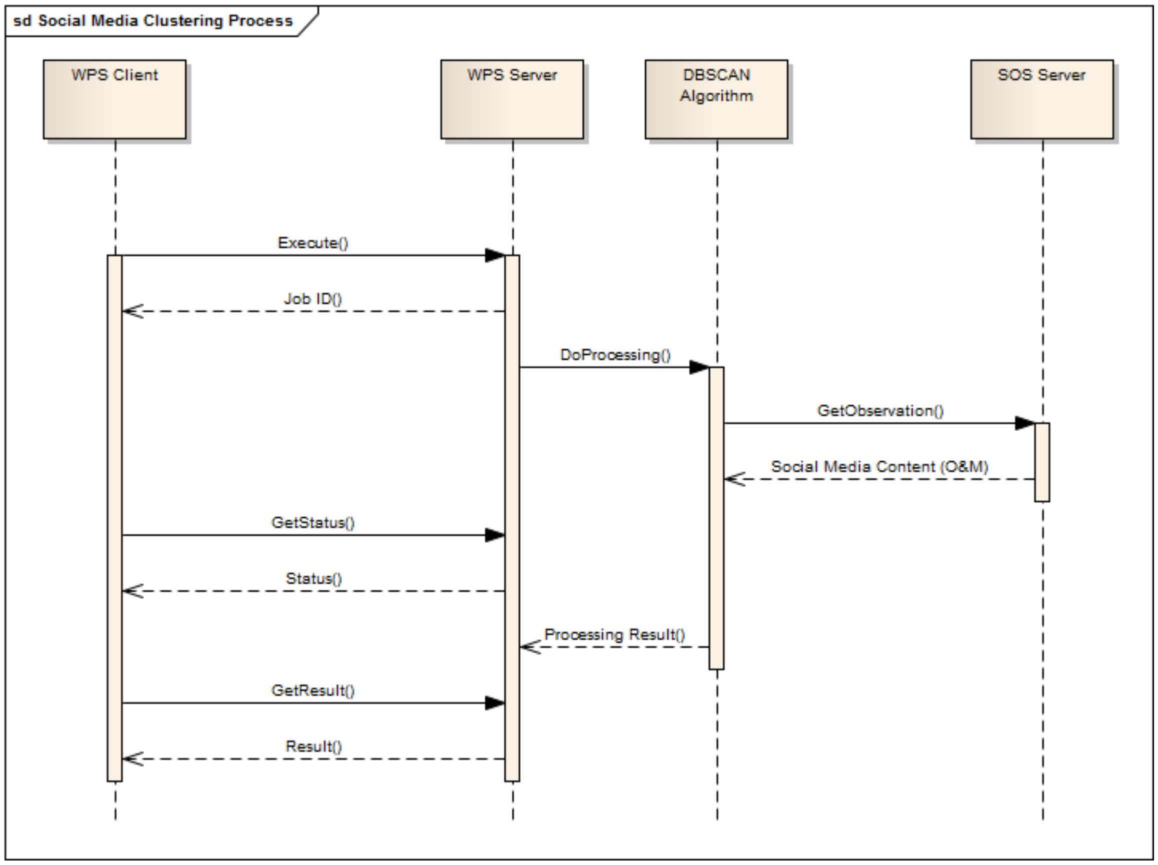


Figure 8 - Sequence diagram of the Social Media process

5.7 Social Media Integration using Linked Data and REST

5.7.1 Overview

Social Media Sources and in particular Volunteered Geographic Information (VGI) has emerged as an important source of up-to-date geo-social information for supporting Emergency Response operations, providing relevant social information (pictures, tweets, narrative, discussion, etc) connected with specific locations and events.

To facilitate the integration of crowd-sourced geosocial information with existing features, POI and gazetteer data used in Emergency Response, Image Matters (IM) implemented a semantic mapping of information from social media sources to the core geospatial ontologies developed by IM during OWS-10. Content such as Tweets, Videos, Articles, Social Events, etc. were mapped as Social Objects to the cross-domain social media ontology SocialML. This ontology, composed of a number of microtheories, has been developed and tested by Image Matters over the last 4 years (partially funded by DARPA) and has been provided as in-kind contribution to this project. The microtheories have been documented in Annex A and the ontology documentation has been published online at the following endpoint: http://ows.usersmarts.com/socialml.

SocialML is an extensible ontological foundation for representing content produced by social media users. It provides a unifying framework that can interoperate with present as well as new and emerging social networks. It is intended to consolidate many existing models (SIOC, Schema.org, Activity Streams) into a single coherent semantic framework. Before SocialML, Social Web sites remained largely isolated silos with their own APIs and data representations (e.g., Google OpenSocial, Facebook Open Graph,Twitter, YouTube, …), generally based upon syntactic and structural representations (mainly JSON and XML) that are poor in semantics, making interoperability between social networks difficult.

To demonstrate the use of SocialML, Image Matters leveraged its semantic middleware platform KnowledgeSmarts^{TM} to *scrape* social media sources into linked data representations based on the SocialML ontologies and the search criteria supported by targeted social media services. This information was then made available using a REST API that can be queried and integrated with other linked data sources used for Emergency Response. The initial focus was on the use of geospatially-enabled social media services such as Twitter and Flickr feeds. The goal of this approach was to demonstrate the ease of integration of different social media sources using linked data technologies and to provide a unified and simpler access API using a REST API for linked data (LD) scraper services and the use of the GEOSPARQL query language on the aggregation of of social content as linked data. This approach represents a significant alternative to representing all social media content as human sensor observations in a Sensor Observation Service (SOS) or WPS. The service could be easily integrated by a third party as a WPS service using a WPS REST API wrapper. The use of linked data standards can simplify discovery and access for “geosocial” data while still providing some of the semantic richness of the original content.

5.7.2 Semantic Social Media Scraper Service

5.7.2.1 Overview

With the large variety of social media APIs providing heterogeneous access to content, it is difficult to integrate information into a coherent framework that supports analysis and leads to informed decisions. Using a common core vocabulary to describe social concepts and activities, we can extend and accommodate the specificities of each API without breaking the coherency of the representation of the social information generated by these sites. The linked data representation of social information using standards such as RDF, OWL, SPARQL and OGC GeoSPARQL provides a consistent, unified representation and means of access. The use of ontologies also enables powerful reasoning capabilities that can relieve some of the cognitive burden on first responders trying to use social media as actionable information for decision support.

The integration of social media content is performed in three layers:

The first layer is composed of the Social Media APIs producing syntax-based social information encoded usually in JSON or XML. This layer is bridged to a semantic layer representation using RDF scraper plug-ins that convert data from a given API to a linked data representation using the SocialML vocabularies and its extensions. A registry of RDF scrape plug-ins keeps track of which native APIs go with which scrapers. The linked data information is then aggregated and stored in a RDF store accessible through either a GeoSPARQL endpoint or a REST API based on the Linked Data API. The GeoSPARQL endpoint can be used to feed different analyses through computational agents (Web Processing Services) or for direct information access. The integrated geosocial information can also be easily linked to other relevant information such as features to augment the value of existing information. The functional architecture of the service is illustrated in Figure 9.

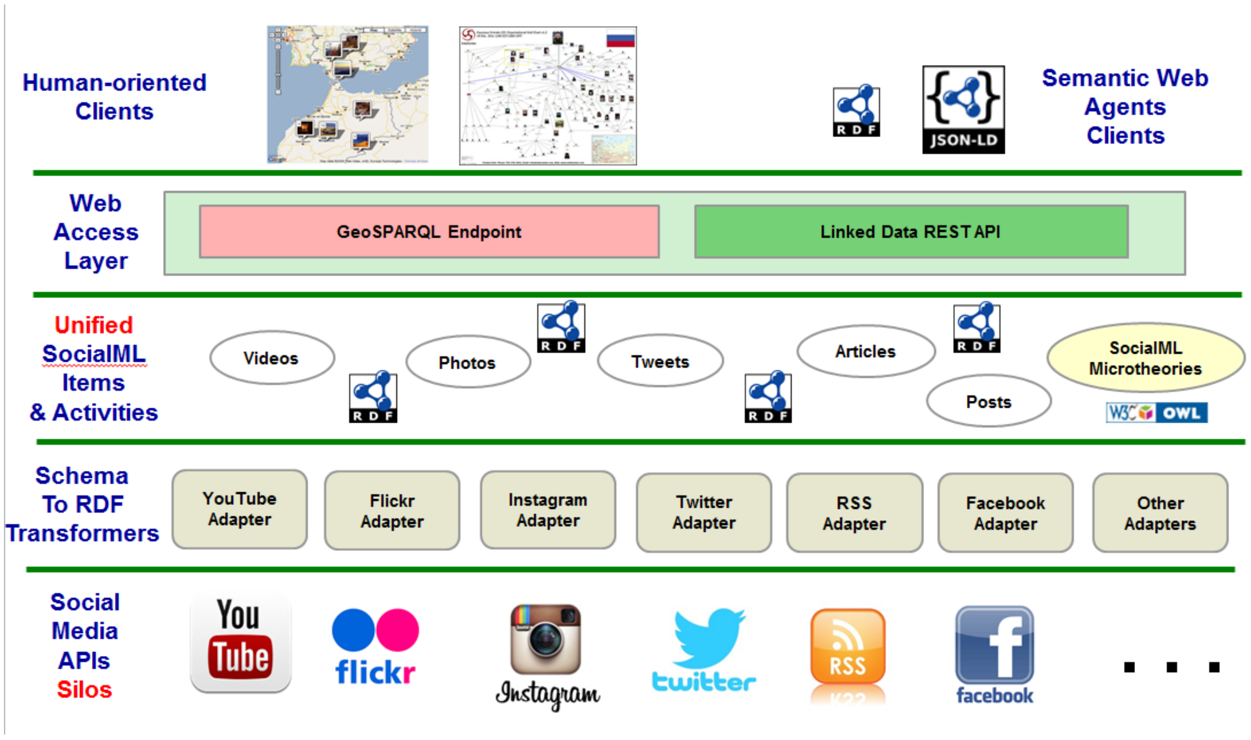


Figure 9 - Functional Architecture of the Semantic Media Scrapper Service

5.7.2.2 REST API

For this testbed, Image Matters implemented a RESTful API to invoke RDF scrapers producing SocialML information from the following web services: Twitter, Flickr and YouTube which produce respectively MicroblogPost, Photo and Video social objects.

The REST API is summarized in the following table.

Table 8 - Overview of the REST API of the Semantic Media Scrapper Service

Method	Endpoint	Description	Response Format
GET	/types	Get the list of the RDF scraper available	TTL,N3,RDF,JSON-LD
GET	/types/{name}	Get the description of the RDF scraper with the given name with its supported parameter	TTL,N3,RDF,JSON-LD
GET	/types/{name}/scrape	Invoke the scraping operation associated with the scraper of the given name[6]	TTL,N3,RDF,JSON-LD

The following MIME types are supported by the REST API and file extensions in URL are supported to make it easy to test manually from a browser (machine should use MIME type negotiation).

Table 9 - MIME Types Supported by the Semantic Media Scrapper Service

MIME	Extension	Description
text/turtle	.ttl	Turtle format
text/n3	.n3	N3 Format
application/rdf+xml	.rdf	RDF/XML format
application/n-triples	.nt	N-Triples format
application/ld+json	.jsonld	JSON-LD format

5.7.2.3 LD Scraper Type Resources

An RDF Scraper Type is a web accessible process accessing a data source (through API call or data document) and produces a linked data representation of the information retrieved from the source. The LD Scraper service uses a pluggable architecture to add new LD Scraper Types (such as Instagram, Vimeo, ISO 19139 documents, etc.). The plugins can be discovered dynamically and the offerings can be accessed through the RESTful API.

[6] The use of the GET operation may be problematic here in terms of REST, however, since it generates content in the knowledgebase and is therefore not idempotent.

5.7.2.3.1 Get the List of available LD scrapers

To get the list of RDF scraper type offerings from the service, the service provides an endpoint that returns a collection of ScraperType instances according the Linked Data Platform API specification.

Request

Endpoint: /types

Method: HTTP Get

The request accepts the following mime types: TTL, N3, RDF/XML, NT and JSON-LD.

Response

Note that this endpoint implements the Linked Data Platform API (using *ldp:BasicContainer* to represent collection in RDF) and the following HTTP header (as standardized in the Linked Data Platform specification[7]).

The following is an example of a response in the Turtle format[8]. The response returns a BasicContainer which refers to the different ScraperType supported by the service (using the standard property *ldp:contains*). The document contains a brief description of each scraper type by providing a title, a description (rdfs:comment) and a unique name that is used in the REST API to get more details about the resource. Note that the uri of each scraper type are resolvable as required by the LDP specification.

Listing 5 - Example of a Response in the Turtle Format

```
Link =  <http://www.w3.org/ns/ldp#BasicContainer>; rel='type',
<http://www.w3.org/ns/ldp#Resource>; rel='type'
@base          <http://ows.usersmarts.com/ldscraper/api/types/> .
@prefix :      <http://ows.usersmarts.com/ldscraper/api/types/> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix scraper: <http://www.knowledgesmarts.com/ontologies/scraper#> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix ldp:   <http://www.w3.org/ns/ldp#> .

:       a              ldp:BasicContainer ;
        rdfs:comment   "Linked Data Platform Container of scraper types"
;
        dct:title      "Scraper types" ;
        ldp:contains   :flickr , :google-geocoder , :youtube , :twitter
.
```

[7] http://www.w3.org/TR/2015/REC-ldp-20150226/

[8] http://www.w3.org/TR/turtle/

```
:flickr  a                scraper:ScraperType ;
        dct:description  "Scraper for scraping Flickr to SocialML" ;
        dct:title        "Flickr Scraper" ;
        scraper:name     "flickr" .

:google-geocoder  a      scraper:ScraperType ;
        dct:description  "Scraper for Google Geocoding API" ;
        dct:title        "Google Geocoding API Scraper" ;
        scraper:name     "google-geocoder" .

:youtube  a              scraper:ScraperType ;
        dct:description  "Scraper for scraping YouTube to SocialML" ;
        dct:title        "YouTube Scraper" ;
        scraper:name     "youtube" .


:twitter  a         scraper:ScraperType ;
        dct:description  "Scraper for tweets matching a query,AOI and
time
                          frame" ;
        dct:title        "TweetScraper" ;
        scraper:name     "twitter" .
```

5.7.2.3.2 Get the description of a Scraper Type

Each LD Scraper Type is defined as instance of *scraper:ScraperType* and is a *rdfs:subclassOf scraper:Scraper*. This mechanism allows classification of scraper types within a knowledge base (RDF store). Each Scraper Type has a unique name used in the REST API. A Scraper Type has the following properties:

Table 10 - Properties of Scraper Types

Name	**Description**	**Datatype**	**Cardinality**
Name	The unique short name for the lens parameter (used in query parameter)	string	1
Label	The display label of the parameter definition	string	0..1 (per language)
description	The description of the parameter definition	string	0..1(per language)

hasParameter	The supported parameters by the scraper type. The parameter name is used as a query parameter name in the REST call to perform the scraping processing /types/{name}/scraper?paramName=paramValue The parameter value needs to be consistent with the valueType defined in the parameter definition.	Parameter	0..n

Each Parameter instance has the following properties:

Table 11 - Properties of Parameters

Name	Description	Datatype	Cardinality
Name	The unique short name for the lens parameter (used in query parameter)	string	1
Label	The display label of the parameter definition	string	0..1 (per language)
description	The description of the parameter definition	string	0..1(per language)
valueType	The type of the value of the parameter. The value type is defined with the uri of a datatype (see xsd datatypes) or a Class.	URI	1
minCount	int	int	0..1
maxCount	The maximum cardinality of the parameter.	int	0..1.
Optional	Inidicates if the parameter is optional	boolean	0..1
defaultValue	Default value of the parameter (could be a literal or a uri of a resource).	datatype value or resource	0..1
predicate	the RDF predicate which maps the parameter value in the instance of this metaclass	URI	1
isLiteral	Indicates if the parameter value is a literal (not needed is XSD namespace is used in value type).	boolean	0..1

Request

Endpoint: /types/{name}

Method: HTTP Get

The request accepts the following mime types: TTL, N3, RDF/XML, NT and JSON-LD.

Response

The following example describes the Twitter Scraper Type. Note that the identifier of the Scraper Type uses a relative path to the @base URI of the document, which is the URL of the REST call. This ensures portability of the API when migrated to different servers. The scraper type name is *twitter* and has multiple parameters. Each parameter corresponds to the query parameter used in the native Twitter API (refer to the Twitter API documentation to get more details).

Listing 6 - Twitter Scraper Type Description

```
@base          <http://ows.usersmarts.com/ldscraper/api/types/twitter>
.
@prefix dc:    <http://purl.org/dc/elements/1.1/> .
@prefix :      <http://localhost:8080/ldscraper/api/types/twitter> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix ks:    <http://www.usersmarts.com/ont/2005/06/ks#> .
@prefix ksd:   <http://www.usersmarts.com/ont/2007/02/ks/descriptor#> .
@prefix scraper: <http://www.knowledgesmarts.com/ontologies/scraper#> .
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix kspf:  <http://www.knowledgesmarts.com/ontologies/pf#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix prm:   <http://www.smartrealm.com/ont/ks/param#> .
@prefix ksfun: <http://www.usersmarts.com/ont/2005/06/ks/functor#> .

:       a                 scraper:ScraperType ;
        rdfs:label        "TweetScraper" ;
        rdfs:subClassOf   scraper:Scraper ;
        dc:description    "Scraper for tweets matching a query, AOI and
time
                           frame" ;
        scraper:name      "twitter" ;
        ksd:hasParameter  [ a               ksd:Parameter ;
                            dc:description  "If specified, returns
tweets with
                                generated before the given date. Date
should be
                                            formatted as YYYY-MM-DD" ;
                            ksd:maxCount    "1"^^xsd:int ;
                            ksd:minCount    "0"^^xsd:int ;
                            ksd:name        "until" ;
                            ksd:predicate
<http://www.knowledgesmarts.com/ontologies/scrapers/twitter#until> ;
                            ksd:valueType   xsd:string
                          ] ;
        ksd:hasParameter  [ a               ksd:Parameter ;
                            dc:description  "If specified, returns
tweets with
                                          status ids less than the
given id." ;
                            ksd:maxCount    "1"^^xsd:int ;
                            ksd:minCount    "0"^^xsd:int ;
                            ksd:name        "maxId" ;
                            ksd:predicate
<http://www.knowledgesmarts.com/ontologies/scrapers/twitter#maxId> ;
```

```
                            ksd:valueType   xsd:long
                          ] ;
        ksd:hasParameter  [ a                 ksd:Parameter ;
                            dc:description    "radius unit of measure
(accepted
                                               values:  km or mi)" ;
                            ksd:defaultValue  "km" ;
                            ksd:maxCount      "1"^^xsd:int ;
                            ksd:minCount      "0"^^xsd:int ;
                            ksd:name          "uom" ;
                            ksd:predicate
<http://www.knowledgesmarts.com/ontologies/scrapers/twitter#uom> ;
                            ksd:valueType     xsd:string
                          ] ;
        ksd:hasParameter  [ a               ksd:Parameter ;
                            dc:description  "longitude of center of
geosearch" ;
                            ksd:maxCount    "1"^^xsd:int ;
                            ksd:minCount    "0"^^xsd:int ;
                            ksd:name        "long" ;
                            ksd:predicate
<http://www.knowledgesmarts.com/ontologies/scrapers/twitter#longitude>
;
                            ksd:valueType   xsd:double
                          ] ;
        ksd:hasParameter  [ a               ksd:Parameter ;
                            dc:description  "radius  geosearch" ;
                            ksd:maxCount    "1"^^xsd:int ;
                            ksd:minCount    "0"^^xsd:int ;
                            ksd:name        "radius" ;
                            ksd:predicate
<http://www.knowledgesmarts.com/ontologies/scrapers/twitter#radius> ;
                            ksd:valueType   xsd:double
                          ] ;
        ksd:hasParameter  [ a               ksd:Parameter ;
                            dc:description  "If specified, returns
tweets with
                            status ids greater than the given id." ;
                            ksd:maxCount    "1"^^xsd:int ;
                            ksd:minCount    "0"^^xsd:int ;
                            ksd:name        "sinceId" ;
                            ksd:predicate
<http://www.knowledgesmarts.com/ontologies/scrapers/twitter#sinceId> ;
                            ksd:valueType   xsd:long
                          ] ;
        ksd:hasParameter  [ a               ksd:Parameter ;
                            dc:description  "Language of the tweet" ;
                            ksd:maxCount    "1"^^xsd:int ;
                            ksd:minCount    "0"^^xsd:int ;
                            ksd:name        "lang" ;
                            ksd:predicate
<http://www.knowledgesmarts.com/ontologies/scrapers/twitter#lang> ;
                            ksd:valueType   xsd:string
                          ] ;
        ksd:hasParameter  [ a               ksd:Parameter ;
                            dc:description  "tweet count" ;
                            ksd:maxCount    "1"^^xsd:int ;
                            ksd:minCount    "0"^^xsd:int ;
                            ksd:name        "count" ;
                            ksd:predicate
<http://www.knowledgesmarts.com/ontologies/scrapers/twitter#count> ;
                            ksd:valueType   xsd:integer
                          ] ;
        ksd:hasParameter  [ a               ksd:Parameter ;
                            dc:description  "Boolean indicating whether
tweets
                                          will be geocoded. Default is
true" ;
                            ksd:maxCount    "1"^^xsd:int ;
                            ksd:minCount    "0"^^xsd:int ;
                            ksd:name        "geocoding" ;
                            ksd:predicate
<http://www.knowledgesmarts.com/ontologies/scrapers/twitter#geocoding>
;
                            ksd:valueType   xsd:boolean
                          ] ;
        ksd:hasParameter  [ a               ksd:Parameter ;
                            dc:description  "query term" ;
                            ksd:maxCount    "1"^^xsd:int ;
                            ksd:minCount    "1"^^xsd:int ;
                            ksd:name        "query" ;
                            ksd:predicate
<http://www.knowledgesmarts.com/ontologies/scrapers/twitter#query> ;
                            ksd:valueType   xsd:string
                          ] ;
        ksd:hasParameter  [ a               ksd:Parameter ;
                            dc:description  "latitude of center of
geosearch" ;
                            ksd:maxCount    "1"^^xsd:int ;
                            ksd:minCount    "0"^^xsd:int ;
                            ksd:name        "lat" ;
                            ksd:predicate
<http://www.knowledgesmarts.com/ontologies/scrapers/twitter#latitude> ;
                            ksd:valueType   xsd:double
                          ] ;
        ksd:hasParameter  [ a               ksd:Parameter ;
                            dc:description  "If specified, returns
tweets with           since the given date. Date should be formatted as
YYYY-MM-DD" ;
                            ksd:maxCount    "1"^^xsd:int ;
                            ksd:minCount    "0"^^xsd:int ;
                            ksd:name        "since" ;
                            ksd:predicate
<http://www.knowledgesmarts.com/ontologies/scrapers/twitter#since> ;
                            ksd:valueType   xsd:string
                          ] .
```

The following scraper types were deployed for Testbed-11:

- **Twitter:** The Twitter Scraper Type supports all the parameters available natively in Twitter API. The API allows tweet search by time range, query text string, and radius from a specific lat/long position. The implementation was based on Twitter4j and the JSON responses were converted to Linked Data representation using SocialML Item ontology which defines **MicroblogPost** type for tweets. The LD representation of each tweet was geocoded using **Google Geocoder** for each tweet that contains hometown information associated with user account. This increases the percentage of tweets with geocoding information.

- **Flickr**:The Flickr Scraper type supports all the parameters available natively in Flickr API. The API allows photos search by text, time range, tags, radius from a lat/long or using Where On Earth Id (woeid) referring to a specific gazetteer location. The implementation was based on Flickr4j librart and the response was converted to Linked Data representation using SocialML Item ontology which defines the concept of **Photo**. The ontology was extended with a Flickr profile to accommodate Flickr specific properties such as the properties pointing to different image resolution (flickr:large_1600_url, flickr:medium_url, etc.). The Place ontology was used to represent the location of the pictures. The geopolitical place hierarchy was built from the response.
- **YouTube**: The YouTube Scraper Type supports a subset of the parameters available natively in the YouTube API. The API allows video search by query term, topicId, date range, and radius from a lat/long position. The implementation used the Google-api-client to access the YouTube API. The response of the API was converted to a Linked Data representation using SocialML Item ontology which defines the concept of Video. The Video class was extended with a YouTube profile to accommodate the notion Channel, which was not defined in SocialML Item.

5.7.2.3.3 Invoke the scraping process for a given scraper type

Request

To invoke the scraping process for a given scraper type, the following endpoint is used:

Endpoint: /types/{name}/scrape.

Method: HTTP Get

The parameter names defined in the scraper type are used as query parameter names in the URL of the REST API. The request accepts the following mime types: TTL, N3, RDF/XML, NT and JSON-LD.

Response

The Response of the scraping process invocation returns a LD representation of the data handled by the scraper in the format requested in the content type of the request. In the case of the Testbed-11, the scrapers used SocialML ontology to represent social objects produced by the different social media sites. The LD Scraper framework does not enforce the use of this ontology; other alternatives ontologies can be used such as SIOC.

5.7.2.3.4 Twitter LD Scraper example

For example, to Find Tweets talking about Flood within 100 km from San Francisco (latitude 37.783333 and longitude -122.416667), the query looks like this:

http://ows.usersmarts.com/ldscraper/api/types/twitter/scrape?query=Crime&lat=37.783333&long=-122.416667&radius=100&unit=km

A sample of the response in TTL from this query is shown below. Note that the conversion of the tweets into Linked Data representation can be accomplished without loss of information.

Listing 7 - Example Response of a Twitter Scraping Process

```
@prefix id:    <http://www.knowledgesmarts.com/ontologies/identifier#>
.
@prefix sioc:  <http://rdfs.org/sioc/ns#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix geo:   <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix count: <http://www.opengis.net/ont/common/count#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix item:  <http://www.socialml.org/ontologies/items#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix social-cnt: <http://www.socialml.org/ontologies/count#> .
@prefix fwprov: <http://www.factweave.com/ontologies/metadata#> .

<http://www.twitter.com/LitterFreeEarth/status/613879796242743296>
        a                      item:MicroBlogPost , item:Tweet ;
        dct:created            "2015-06-25T01:22:03Z"^^xsd:dateTime ;
        dct:creator            <http://www.twitter.com/LitterFreeEarth>
;
        dct:identifier         "613879796242743296"^^xsd:long ;
        dct:language           "en" ;
        dct:source             <http://www.twitter.com> ;

        <http://www.knowledgesmarts.com/ontologies/place#hasPlace>
                <http://api.twitter.com/1/geo/id/5a110d312052166f> ;
        <http://www.knowledgesmarts.com/ontologies/place#location>
                <geo:37.80669245,-122.40456051> ,
<geo:53.35701770000001,-2.0853567> ;
        count:hasCount
<http://www.twitter.com/LitterFreeEarth/status/613879796242743296/retwe
etCount/06-24-2015-23-23-06.396> ,
<http://www.twitter.com/LitterFreeEarth/status/613879796242743296/favor
iteCount/06-24-2015-23-23-06.396> ;
        item:hasLink           <https://instagram.com/p/4VWE0vo49G/> ;
        item:hashtag           "Alcatraz" , "plasticbottle" ;
        item:text              "#Alcatraz may no longer be a prison,
but a crime has been committed on the island. #plasticbottle…
https://t.co/5FCJsum4LE" ;
        item:url
<http://www.twitter.com/LitterFreeEarth/status/613879796242743296> .
```

```
<https://instagram.com/p/4VWE0vo49G/>
        a             item:Link ;
        item:shortUrl  "https://t.co/5FCJsum4LE" ;
        item:url       "https://instagram.com/p/4VWE0vo49G/" .

<geo:37.80669245,-122.40456051>
        a          geo:Point ,
<http://www.opengis.net/ont/spatial/geometry/point#Point> ;

<http://www.knowledgesmarts.com/ontologies/spatial/geometry#asWKT>
                "POINT(37.80669245 -
122.40456051)"^^<http://www.opengis.net/ont/geosparql#wktLiteral> ;
        geo:lat   "37.80669245"^^xsd:double ;
        geo:long  "-122.40456051"^^xsd:double .

<http://api.twitter.com/1/geo/id/5a110d312052166f>
        a       <http://www.knowledgesmarts.com/ontologies/place#Place>
;
        <http://www.knowledgesmarts.com/ontologies/place#countryName>
                "United States" ;
        <http://www.knowledgesmarts.com/ontologies/place#placeName>
                "San Francisco, CA" .

<http://www.twitter.com/LitterFreeEarth>
        a                        sioc:UserAccount ;
        dct:created              "2009-03-20T03:10:28Z"^^xsd:dateTime
;
        dct:description          "Saving the planet, one piece of
litter at a time. Cathleen Murphy http://t.co/jiYdPg3vqm" ;
        dct:language             "en" ;
        count:hasCount
<http://www.twitter.com/LitterFreeEarth/followingCount/06-24-2015-23-
23-06.396> , <http://www.twitter.com/LitterFreeEarth/followerCount/06-
24-2015-23-23-06.396> ,
<http://www.twitter.com/LitterFreeEarth/statusCount/06-24-2015-23-23-
06.396> ;
        social-cnt:numFollowers   "2735"^^xsd:int ;
        social-cnt:numFollowings  "1950"^^xsd:int ;
        social-cnt:numStatuses    "7444"^^xsd:int ;
        foaf:accountName          "LitterFreeEarth" ;
        foaf:based_near           "Planet Earth" ;
        foaf:depiction
<http://pbs.twimg.com/profile_images/378800000054889560/37c84077c09d558
44afeddabc415f889_normal.jpeg> ;
        foaf:name                 "Litter Free Planet" .


<http://www.twitter.com/LitterFreeEarth/followingCount/06-24-2015-23-
23-06.396>
        a             social-cnt:FollowingCount ;
        rdfs:label     "Friend Count for user 25448508 at 2015-06-
24T23:23:06.396-04:00" ;
        dct:date       "2015-06-25T03:23:06.396Z"^^xsd:dateTime ;
        count:count    "1950"^^xsd:int ;
        count:countOf  <http://www.twitter.com/LitterFreeEarth> .
```

```
<http://www.twitter.com/LitterFreeEarth/followerCount/06-24-2015-23-23-
06.396>
        a               social-cnt:FollowerCount ;
        rdfs:label      "Follower Count for user 25448508 at 2015-06-
24T23:23:06.396-04:00" ;
        dct:date        "2015-06-25T03:23:06.396Z"^^xsd:dateTime ;
        count:count     "2735"^^xsd:int ;
        count:countOf   <http://www.twitter.com/LitterFreeEarth> .

<http://www.twitter.com/LitterFreeEarth/statusCount/06-24-2015-23-23-
06.396>
        a               social-cnt:StatusCount ;
        rdfs:label      "Status Count for user 25448508 at 2015-06-
24T23:23:06.396-04:00" ;
        dct:date        "2015-06-25T03:23:06.396Z"^^xsd:dateTime ;
        count:count     "7444"^^xsd:int ;
        count:countOf   <http://www.twitter.com/LitterFreeEarth> .

<http://www.twitter.com/LitterFreeEarth/status/613879796242743296/retwe
etCount/06-24-2015-23-23-06.396>
        a               social-cnt:RetweetCount ;
        rdfs:label      "Retweet Count for
http://www.twitter.com/LitterFreeEarth/status/613879796242743296 at
2015-06-24T23:23:06.396-04:00" ;
        dct:date        "2015-06-25T03:23:06.396Z"^^xsd:dateTime ;
        count:count     "0"^^xsd:int ;
        count:countOf
<http://www.twitter.com/LitterFreeEarth/status/613879796242743296> .

<http://www.twitter.com/LitterFreeEarth/status/613879796242743296/favor
iteCount/06-24-2015-23-23-06.396>
        a               social-cnt:FavoriteCount ;
        rdfs:label      "Favorite Count for tweet
http://www.twitter.com/LitterFreeEarth/status/613879796242743296 at
2015-06-24T23:23:06.396-04:00" ;
        dct:date        "2015-06-25T03:23:06.399Z"^^xsd:dateTime ;
        count:count     "0"^^xsd:int ;
        count:countOf
<http://www.twitter.com/LitterFreeEarth/status/613879796242743296> .

<geo:53.35701770000001,-2.0853567>
        a           geo:Point ,
<http://www.opengis.net/ont/spatial/geometry/point#Point> ;
        rdfs:label  "Planet Earth" ;

<http://www.knowledgesmarts.com/ontologies/spatial/geometry#asWKT>
                "POINT(53.35701770000001 -
2.0853567)"^^<http://www.opengis.net/ont/geosparql#wktLiteral> ;
        geo:lat     "53.35701770000001"^^xsd:double ;
        geo:long    "-2.0853567"^^xsd:double .
```

5.7.2.3.5 Flickr LD Scraper example

Similarly, the Flickr Scraper Type can be invoked the same way by using the Flickr parameter names as query parameters in the URL.

For example to get the flood pictures (tags flood and inundation) of San Francisco (Where On Earth ID = 2487956) that happened after Dec.12, 2014, the query would look like this:

http://ows.usersmarts.com/ldscraper/api/types/flickr/scrape?tags=flood,inundation%20&woe_id=2487956&min_taken_date=2014-12-01

The response in TTL looks like this:

Listing 8 - Example Response of a Flickr Scraping Process

```
@prefix flickr: <http://www.knowledgesmarts.com/ontologies/flickr#> .
@prefix geosparql: <http://www.opengis.net/ont/geosparql#> .
@prefix point: <http://www.opengis.net/ont/spatial/geometry/point#> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix item:  <http://www.socialml.org/ontologies/items#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix geoplanet:
<http://www.knowledgesmarts.com/ontologies/geoplanet#> .
@prefix wgs84: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix geom:  <http://www.opengis.net/ont/spatial/geometry#> .
@prefix place: <http://www.opengis.net/ont/place#> .

<https://www.flickr.com/photos/fredarmitage/15377842444/>
        a                         item:Photo ;
        dct:created               "2014-12-11T13:40:30Z"^^xsd:dateTime ;
        dct:creator
<http://www.flickr.com/people/58746120@N00> ;
        dct:dateAccepted          "2014-12-11T21:17:43Z"^^xsd:dateTime ;
        dct:description           "Found an old waterproof video camera
with a 5mp camera. What better tool to use today 12/11/2014 in the big
storm?" ;
        dct:modified              "2015-04-30T08:12:08Z"^^xsd:dateTime ;
        dct:subject               "flash flood" , "wind" , "california"
, "12/11/2014 storm" , "flooding" , "flood" , "12/11/2014" , "sewage" ,
"overflowing" , "overflow" , "inundations" , "inundation" , "storm" ,
"weather" , "street" , "water" , "fall" , "rainfall" , "rain" , "san
francisco" , "sf" , "drought" , "feet in water" , "rain storm" , "wind
storm" , "pineapple express" , "pedestrian" , "crossroad" , "flooded
crossroad" , "san francisco storm" , "san francisco rain" , "rainy" ,
"flooded" ;
        dct:title                 "Heard in the street today, I quote:
\"Drought? What #drought?\"" ;
        flickr:large_1600_url
<https://farm8.static.flickr.com/7527/15377842444_3cf3dca267_h.jpg> ;
        flickr:large_2048_url
<https://farm8.static.flickr.com/7527/15377842444_3cf3dca267_k.jpg> ;
```

```
        flickr:large_square_url
<https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_q.jpg> ;
        flickr:large_url
<https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_b.jpg> ;
        flickr:medium_640_url
<https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_z.jpg> ;
        flickr:medium_800_url
<https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_c.jpg> ;
        flickr:medium_url
<https://farm8.staticflickr.com/7527/15377842444_3cf3dca267.jpg> ;
        flickr:photoId           "15377842444" ;
        flickr:small_320_url
<https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_n.jpg> ;
        flickr:small_square_url
<https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_s.jpg> ;
        flickr:small_url
<https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_m.jpg> ;
        place:hasLocation        [ a                  place:Place ;
            place:adminLevel1
<http://where.yahooapis.com/v1/place/2347563> ;
            place:adminLevel2
<http://where.yahooapis.com/v1/place/12587707> ;
            place:country
<http://where.yahooapis.com/v1/place/23424977> ;
            place:locality
<http://where.yahooapis.com/v1/place/2487956> ;
            foaf:based_near    [ a           wgs84:Point , point:Point
                     geom:asWKT  "POINT(37.78455352783203 -
122.4045181274414)"^^geosparql:wktLiteral ;
                     wgs84:lat   "37.78455352783203"^^xsd:double ;
                     wgs84:long  "-122.4045181274414"^^xsd:double
                                 ]
          ] ;
        item:originalURL
<https://farm8.staticflickr.com/7527/15377842444_ff31e38086_o.jpg> ;
        item:thumbnailURL
<https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_t.jpg> ;
        item:url
<https://www.flickr.com/photos/fredarmitage/15377842444/> .

<http://www.flickr.com/people/58746120@N00>
        a                            foaf:OnlineAccount ;
        foaf:accountName             "Frédéric Poirot" ;
        foaf:accountServiceHomepage  <http://www.flickr.com> ;
        foaf:name                    "Frédéric Poirot" .

<http://where.yahooapis.com/v1/place/2487956>
        a                  place:Place ;
        flickr:placeId     "7.MJR8tTVrIO1EgB" ;
        geoplanet:woeid    "2487956" ;
        place:adminLevel1
<http://where.yahooapis.com/v1/place/2347563> ;
        place:adminLevel2
<http://where.yahooapis.com/v1/place/12587707> ;
```

```
        place:country
<http://where.yahooapis.com/v1/place/23424977> ;
        place:placeName    "San Francisco" ;
        place:placeType
<http://where.yahooapis.com/v1/placetype/locality> .

<http://where.yahooapis.com/v1/placetype/locality>
        a                    place:PlaceType ;
        place:placeTypeName  "locality" .

<http://where.yahooapis.com/v1/place/12587707>
        a                  place:Place ;
        flickr:placeId     ".7sOmlRQUL9nK.kMzA" ;
        geoplanet:woeid    "12587707" ;
        place:adminLevel1
<http://where.yahooapis.com/v1/place/2347563> ;
        place:country
<http://where.yahooapis.com/v1/place/23424977> ;
        place:placeName    "San Francisco" ;
        place:placeType
<http://where.yahooapis.com/v1/placetype/adminLevel2> .

<http://where.yahooapis.com/v1/placetype/adminLevel2>
        a                    place:PlaceType ;
        place:placeTypeName  "adminLevel2" .

<http://where.yahooapis.com/v1/place/2347563>
        a                place:Place ;
        flickr:placeId   "NsbUWfBTUb4mbyVu" ;
        geoplanet:woeid  "2347563" ;
        place:country    <http://where.yahooapis.com/v1/place/23424977>
;
        place:placeName  "California" ;
        place:placeType
<http://where.yahooapis.com/v1/placetype/adminLevel1> .

<http://where.yahooapis.com/v1/placetype/adminLevel1>
        a                    place:PlaceType ;
        place:placeTypeName  "adminLevel1" .

<http://where.yahooapis.com/v1/place/23424977>
        a                place:Place ;
        flickr:placeId   "nz.gsghTUb4c2WAecA" ;
        geoplanet:woeid  "23424977" ;
        place:placeName  "United States" ;
        place:placeType
<http://where.yahooapis.com/v1/placetype/country> .

<http://where.yahooapis.com/v1/placetype/country>
        a                    place:PlaceType ;
        place:placeTypeName  "country" .
```

5.7.2.3.6 YouTube LD Scraper example

Find Videos of Flooding in San Francisco within 100 km from San Francisco (lat=37.783333&long=-122.416667)

http://ows.usersmarts.com/ldscraper/api/types/youtube/scrape?location=37.783333%2C-122.416667&locationRadius=100km&query=flood

Listing 9 - Example Response of a Youtube Scraping Process

```
@prefix dct:   <http://purl.org/dc/terms/> .
@prefix item:  <http://www.socialml.org/ontologies/items#> .
@prefix ytube: <http://www.knowledgesmarts.com/ontologies/youtube#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .

<https://youtu.be/P1mB2FVQcgs>
        a                    item:Video ;
        dct:dateSubmitted    "2015-03-14T14:15:22.000Z"^^xsd:dateTime ;
        dct:description      "On December 11, 2014, drought stricken San
Francisco Bay Area, California was hit with a strong rain storm that
produced flash flooding in several areas." ;
        dct:title            "Flash Flood Engulfs My Neighborhood In
Minutes" ;
        ytube:channel
<http://www.youtube.com/channels/UC6mUFwnE4A5AHcv0ihiRduw> ;
        item:embedUrl        <http://www.youtube.com/embed/P1mB2FVQcgs> ;
        item:thumbnailUrl
<https://i.ytimg.com/vi/P1mB2FVQcgs/default.jpg> ;
        item:url             <https://youtu.be/P1mB2FVQcgs> .

<http://www.youtube.com/channels/UC6mUFwnE4A5AHcv0ihiRduw>
        a                  ytube:Channel ;
        dct:title          "GottaBme4me2bFree" ;
        ytube:channelOf
<http://www.youtube.com/channels/UC6mUFwnE4A5AHcv0ihiRduw> .

<https://youtu.be/IMUvPoczpaU>
        a                    item:Video ;
        dct:dateSubmitted    "2014-12-11T21:51:27.000Z"^^xsd:dateTime ;
        dct:description      "Sonoma County Home by Santa Rosa Creek hit
by Flash Flood." ;
        dct:title            "Sonoma County Home hit by Flash Flood" ;
        ytube:channel
<http://www.youtube.com/channels/UCQTcZR01PoR_atNA5-bU2Kg> ;
        item:embedUrl        <http://www.youtube.com/embed/IMUvPoczpaU> ;
        item:thumbnailUrl
<https://i.ytimg.com/vi/IMUvPoczpaU/default.jpg> ;
        item:url             <https://youtu.be/IMUvPoczpaU> .

<http://www.youtube.com/channels/UCQTcZR01PoR_atNA5-bU2Kg>
        a                  ytube:Channel ;
        dct:title          "crunnings1" ;
        ytube:channelOf
<http://www.youtube.com/channels/UCQTcZR01PoR_atNA5-bU2Kg> .
```

6 Accomplishments

As a result of the activities described in this report, several goals were achieved.

First, we reviewed of different social media platforms regarding their suitability for supporting a service-based emergency response information scenario. The findings of this review are summarized in Section 5.2. Based on these findings we selected the following platforms (Flickr, Instagram, YouTube and Twitter) for further work within the Testbed.

As a next step we were able to define two alternative information models and associated toolsets for representing social media content based on the findings of the OGC Testbed-10.

- The first model is based on modeling social media content as observations. This resulted in an O&M encoding which was subsequently implemented by an SOS server that served as data access interface and discovery tool for the selected social media platforms.
- The second model is based on modeling social media content as social objects on which users of social media perform activities (posting, commenting, rating, etc…). Objects and activities were represented using SocialML ontologies provided by Image Matters, which model social items as linked data, without loss of information and with the ability to link social information to other domains (feature of interest for example such as Incident) using linked data standards (RDF, OWL, Linked Data Platform, SPARQL, GeoSPARQL).

To fill the SOS 2.0 for social media with data, three harvesters for social media content were developed. While the harvesters for Flickr and Twitter are updates of the results achieved in the OGC Testbed-10 (in these cases updates to accommodate changes in the social media APIs were needed) the harvester for Instagram was a new development.

To demonstrate the linked data approach for integrating social media information, a RESTful Linked Data Scraper Service was implemented by Image Matters to demonstrate the use of a REST API, Linked Data Platform (LDP) specifications, and SocialML ontologies. Three LD Scraper types were implemented: Twitter, YouTube and Flickr.

To further analyze social media content, especially to detect clusters of relevant information and thus to derive information about critical events, a WPS server was developed offering interoperable access to the DBSCAN clustering algorithm encapsulated as a WPS process.

Finally, a client application integrating the data sources described in this report as well as processing tools encapsulated through a WPS server was provided.

7 Lessons Learned

Our activities during the Testbed showed that OGC standards, especially those of the Sensor Web Enablement framework, are a viable basis for handling social media content within spatial data infrastructures and in application contexts such as emergency response.

The OGC SOS 2.0 interface standard offers an interoperable approach to discover, query, and access social media content. A lot of content of social media platforms is based on human perception as thus can be considered as human observations, the O&M 2.0 model and encoding offers a straightforward approach for exchanging the content of platforms such as Flickr, Instagram, and Twitter. The SOS 2.0 interface offering direct query parameters for properties typical for any kind of observation ensures a high level of interoperability which goes beyond the level that can be achieved with WFS servers (e.g. through explicit query fields such as procedure, observed property, etc.).

Furthermore, the transactional operations of the SOS 2.0 interface facilitate the publication of social media content through SOS servers. They allow for the quick creation of harvesting mechanisms that query social media APIs and convert the discovered content into an O&M 2.0 representation.

With regard to O&M 2.0 we observed that guidance may be helpful to agree on a common way how to encode additional content of social media observations (e.g. keywords or tags) within an O&M observation. The use of controlled vocabularies may fill some of these gaps, but additional forms of analysis will be needed to more specifically delineate the phenomena, features of interest, and results that can be gleaned from human sensors.

The linked data approach for social media integration has been investigated and implemented using REST and linked data standards and shown to be a viable solution for semantic integration of social information. The view of social content as social objects also generalizes their use, as not all social objects are the results of human observations. For example, a map is a social object on which users can perform different actions such as rating, commenting, and sharing. The linked data approach might provide a higher level of interoperability than standards that are mostly based on structural and syntactical representations of information. The ability to represent varied social information using ontologies and linking them to other domains as linked data through REST APIs provides an opportunity to get more value from existing information (by way of linking and machine reasoning). A linked data representation of social information can be easily aggregated with other RDF stores and accessed using standard query languages (SPARQL/GeoSPARQL) or linked data APIs. In this context, further experience will be helpful to identify the approaches best suited for particular usage scenarios.

Issues identified during the testbed include the low proportion of georeferenced content on most platforms. This reduces the value of the discovered content as it is not possible to associate the information with specific locations and/or features. Further constraints such as social media API limitations on querying for historic data also reduce the amount of data that can be used.

Finally, intellectual property right restrictions limit the applicability of social media content in many (commercial) scenarios. Here, further work will be necessary, to achieve agreements with operators of social media platforms to contribute their content in emergency situations.

8 Recommendations and Future Work

Based on our experiences gained during the OGC Testbed-11 we provide the following recommendations as future work:

For Social Media Integration using SOS:

- Create a profile describing how to handle social media content within the OGC Sensor Web Enablement (SWE) standards based on the findings of OGC Testbed-10 and -11. This should ideally be developed as an OGC Best Practice Document.
- The focus of the SOS 2.0 for Social Media was on the provision of O&M-encoded social media content. However, the provision of additional metadata (e.g. information about the observer encoded in SensorML) may help to assess the suitability, quality, and reliability of information discovered on social media platforms.
- Further analysis should be performed to determine the legal conditions under which the content of social media platforms may be used by (non-profit or commercial) tools for emergency response.
- Within the WPS 2.0 for Social Media instance we are applying an algorithm for detecting clusters within the published social media content. If such algorithms are to be used to automatically detect crisis events, the reliability of social media content needs to be further investigated and means of evaluating it developed. For example scenarios such as the one described in http://www.nytimes.com/2015/06/07/magazine/the-agency.html?smprod=nytcore-ipad&smid=nytcore-ipad-share&_r=0 can lead to false alarms.
- Additional work on trend analysis should be carried out. The prototypical workflow developed in this testbed did not take temporal changes into account. A cluster analysis could be applied on floating time windows which could be intuitively visualized in the client application.

Recommended future work for social media integration as linked data includes:

- Further investigation of the use of the standard W3C Linked Data Platform API to perform CRUD operations on linked data representations
- Integration of more social media APIs with similar social objects (for example Vimeo, Paranomio, Instagram) to test the search over social objects of similar types from diverse sources
- Support of asynchronous scraping tasking with scheduling and aggregation of results in an RDF store for subsequent analysis
- Investigation of the use of GeoSPARQL and other SPARQL extensions with social media to support social analytics.
- Investigation of process descriptions as linked data (generalizing the scraper type descriptions for any processes) and use of RESTful forms of WPS services.
- Integration and linking of social content linked data with data from other domains (Incident, Feature of Interest etc.).

- Definition of Best Practices for linked data integration of social media content.

9 References

Hawelka, Bartosz; Izabela Sitko; Euro Beinat; Stanislav Sobolevsky; Pavlos Kazakopoulos & Carlo Ratti (2014): Geo-located Twitter as proxy for global mobility patterns, Cartography and Geographic Information Science, 41:3, 260-271, DOI: 10.1080/15230406.2014.890072

Jurgens, David; Tyler Finnethy, James McCorriston, Yi Tian Xu & Derek Ruths (2015): Geolocation Prediction in Twitter Using Social Networks: A Critical Analysis and Review of Current Practice, The 9th International Conference on Weblogs and Social Media (ICWSM). Online at: http://cs.mcgill.ca/~jurgens/docs/jurgens-et-al_icwsm-2015.pdf

Speicher, Steve; John Arwe; Ashok Malhotra. Linked Data Platform 1.0. 26 February 2015. W3C Recommendation. URL: http://www.w3.org/TR/ldp/

Annex A

SocialML Microtheories

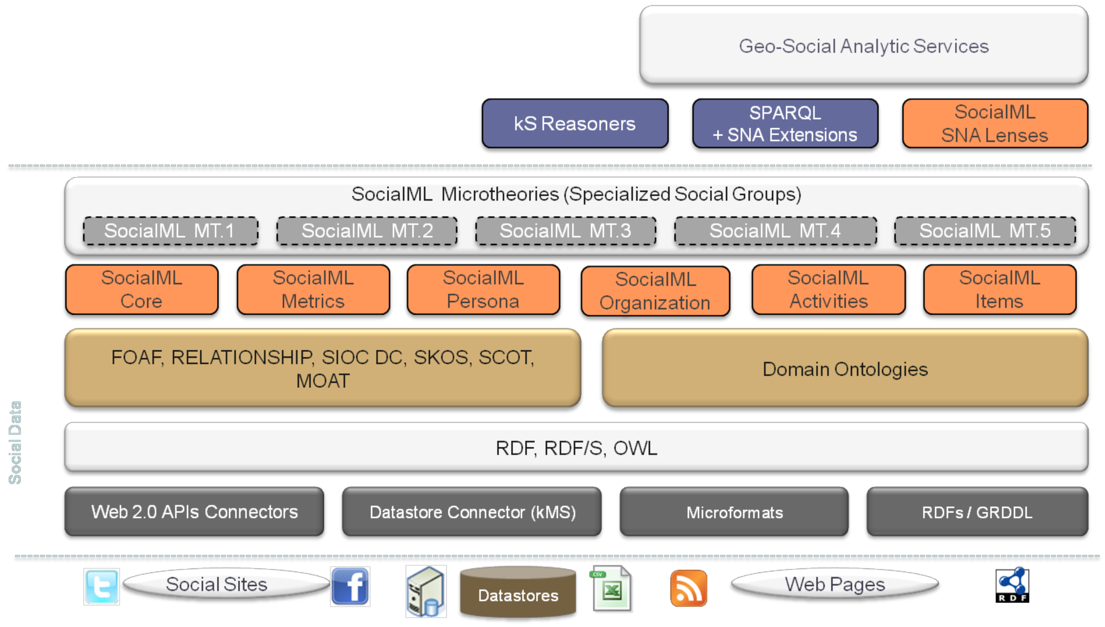
A.1 SocialML Microtheories Overview

The rapid emergence of Web 2.0 social sites quickly led to an array of new social objects, activities and personas that could better accommodate a broader scoped social experience. This social experience surpassed simple tagging operations found in early social bookmarking sites (delici.o.us) and led Image Matters to the realization that a common model was needed across various types of social media to capture and integrate social information, to build a richer Persona description of an individual or community. This persona includes profile information, cognitive characteristics (skills, beliefs, expertise, interests, goals), relationships, interactions with other members of communities and social objects, rich descriptions of social objects, and influences and roles in different communities. Image Matters, under a DARPA contract, created a comprehensive, extensible and semantic framework for Social Media and networks (SocialML). This framework may be used to model Social Media information. It also serves as the basis for performing semantic-based geo-social analytics and overcoming interoperability barriers between Social Media sites.

Absent a common model to describe the heterogeneity of social media, social web sites remain isolated silos limited by their own APIs and data representations (e.g., Google OpenSocial, Facebook Open Graph, Flickr, Twitter, etc.). These APIs are generally based upon syntactic and structural representations (mainly JSON and XML) that are semantically deficient, making interoperability between social networks difficult. Other current efforts to unify Web 2.0 information are not semantic-based (ActivityStreams, Atom), further impeding efforts to automate information reasoning and fusion. Scattered attempts to semantically formalize social information via various specifications (e.g. Friend of a Friend (FOAF), SIOC, and GoodRelations) have failed to unify the collective set of social/se- mantic community requirements.

New social sites and API versions pop up regularly. A unifying framework is desirable to unify and bridge social media models in a decentralized way, and inter-operate with other new and emerging social networks. To fill this gap, Image Matters created a family of microtheories for social networks called SocialML.

SocialML bridges this gap by providing a set of core ontologies for representing Social Networks, Persona, social activities and objects, organizations, social relationships and social network analysis metrics. The diagram below shows in orange the SocialML modules that have been added to our semantic platform. It is clear that SocialML ontologies have the potential to have a tremendous impact on the interoperability of existing Web 2.0 social networks, as well as their integration with the Linked Data Web (Web 3.0). As a standard, SocialML would benefit the Web 2.0 community at large by removing the barriers of interoperability between various social media and social network services.



A.2 Design Approach

This section outlines some the key principles used to design the SocialML ontologies.

A.2.1 Minimal Ontological Commitment

The modular design of the ontologies follows the principle of making a minimal ontological commitment (MOC) to the nature of concepts and of relationships between concepts. As explained by Thomas Gruber, an ontology should require the *minimal ontological commitment sufficient to support the intended knowledge sharing activities*. An ontology should make as few claims as possible about the world being modeled, allowing the parties committed to the ontology freedom to specialize and instantiate the ontology as needed (which is often called ontology profile).

Opting for such a minimal approach is made dramatically easier by the vocabulary extension mechanisms offered natively by Semantic Web technology. Applications that require more constrained behavior may define compatible extensions to OWL or SKOS. For example, modelers may coin sub-classes and sub-properties of OWL or SKOS properties, or associate those properties with specific formal axioms. By making a minimal ontological commitment, the ontologies can be applied and reused across multiple Communities of Interests (COIs), thus increasing the rate of wide-spread adoption.

Failure to adhere to the principles minimal ontological commitment result in ontologies that cannot be reused or integrated across multiple communities of interest, and will result in greater resources being required to mediate between similar domain-specific ontologies (i.e. how many different ways can we model features).

A.2.2 Modularization of ontologies

Quoting Stuckenschmidt and Klein (Stuckenschmidt & Klein, 2004), "ontologies that contain thousands of concepts cannot be created and maintained by a single person". Modularization helps designers manage complexity by reducing the size of the design problem (Stuckenschmidt, Parent, & Spaccapietra, 2009). Ontologies must be small enough to easily comprehend, and later either integrate these modules into a final repository or build the relationships among modules that support interoperability. This is a typical application of the divide-and-conquer principle.

Modularization also provides a way to keep performance of ontology services at an acceptable level. Performance concerns may be related to query processing techniques, reasoning engines and ontology modeling and visualization tools. Reasoning engines currently available are performing well on small-scale ontologies, with performance degrading rapidly as the size of the ontology increases. Keeping ontologies small is one way to avoid the performance loss, and modularization is a way to replace an ontology that tends to become oversized by smaller subsets. Modularization fulfills the performance goal if, whenever a query has to be evaluated or an inference to performed, this can be done by looking at just few modules, rather than exploring the whole ontology (Stuckenschmidt et al., 2009).

A.2.3 Reusability of ontologies

Reusability is a well-known goal in software engineering. Just as in software code development the creation of large, monolithic ontologies is counter to principles of sound ontologies. Using composition principles, small, reusable ontologies can be used in domain-specific combinations. Reusability emphasizes the need for rich mechanisms to describe module, in a way that maximizes the chances for modules to be understood, selected and used by other services and applications.

A.2.4 Understandability

An obvious prerequisite is the ability to use ontology to understand its content. Whether the content is shown in visual or textual format, understanding is easier if the ontology is small (a module). Small ontologies are undoubtedly preferable if the user is a human being. Size, however is not the only criterion that influences understandability. More importantly, the way it is structured contributes to improving or decreasing understandability.

A.3 Microtheories Overview

The microtheories of SocialML are organized according the following categories:

- Items
- Activities
- Persona
- Relationships
- Social Groups

- Social Network Analysis (SNA)

A.3.1 Social Items Microtheories

Social Networks are built upon one or more Social Objects, on which different activities (or actions) can be performed. For example, video social networks are formed around Video, Channel and Comments, upon which posting, viewing, rating and commenting activities can be performed. Flickr has Photo objects as the central social object of interest. Our core ontology for representing SocialML Objects defines the entity of *Item*, which is equivalent to the `sioc:Item` concept. Semantically-Interlinked Online Communities (SIOC), is an attempt to link online community sites, to use Semantic Web technologies to describe the information that communities have about their structure and contents, and to find related information and new connections between content items and other community objects. SIOC is based around the use of machine-readable information provided by these sites. Our SocialML Item ontology extends SIOC by providing a richer list of items types and also draws from schema.org.

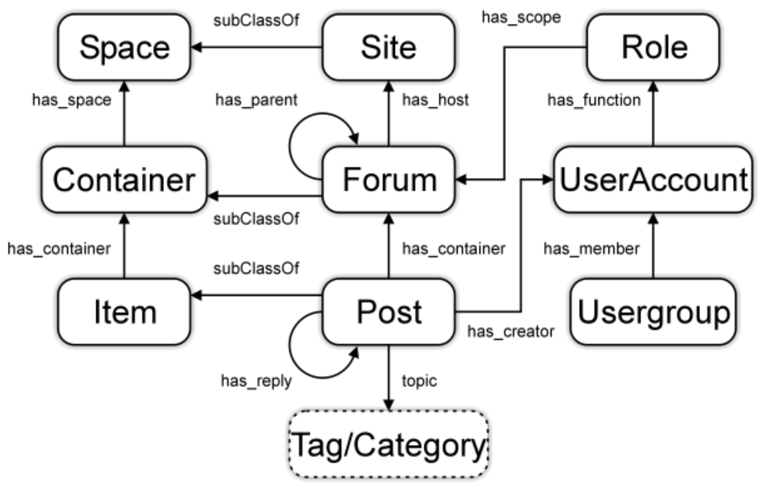


Figure 10 - SIOC Model

The Social Items and Activities Ontology is part of SocialML and used to model and classify all social objects. The social items `item:Book`, `item:Media` (`item:Audio`, `item:Image`, `item:Video`), and `item:Post` are all characterized as being an `item:CreativeWork`. The most important social items include:

- **Post**: A Post is a message posted by a `UserAccount` to a `Forum`. A series of `Posts` may be threaded if they share a common subject and are connected by reply or by date relationships. `Posts` will have content and may also have attached files, which can be edited or deleted by the `Moderator` of the `Forum` that contains the `Post`.

- **Article** - Articles generally consist of paragraphs of text, in some cases incorporating embedded media such as photos and inline hyperlinks to other resources.
- **Comment** - The 'comment' object type represents a textual response to another object.
- **Note** - The 'note' Object type represents short-form text messages. This Object type is intended for use in 'micro-blogging' and in systems where users are invited to publish short, often plain-text messages whose useful lifespan is generally shorter than that of an article or weblog entry. A note is similar in structure to an article, but it does not have a title and its body tends to be shorter. Applications will often display the entire content of a note in an activity stream UI, whereas they MAY display only the title or the title and summary for a weblog entry.
- **Status** - The 'status' Object type represents a human-readable update of the author's situation, mood, location or other status. A status is similar in structure to a note, but carries the additional meaning that the content is primarily describing something its author is doing, feeling or experiencing. A consumers MAY consider the content of the most recent status object it encountered to be the user's current status, unless the most recent status update is old. When a status becomes too old is not defined by this specification.
- **Media** - consists of `item:Audio`, `item:Image`, and `item:Video`.
- **Bookmark** - The 'bookmark' Object type represents a pointer to some Uniform Resource Locator (URL) – typically a web page. In most cases, a bookmark is specific to a given user and contains Metadata chosen by that user. Bookmark Objects are similar in principle to the concept of bookmarks or favorites in a web browser. A bookmark represents a pointer to the URL, not the URL or the associated Resource itself.

Figure 11 and Figure 4 show the Item and Container hierarchy in the core Item ontology.

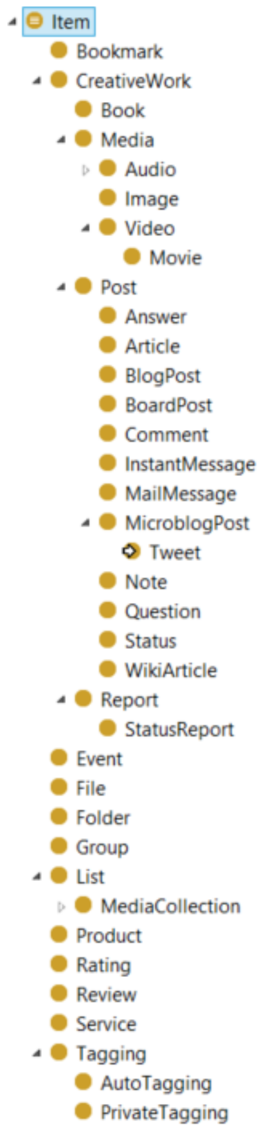


Figure 11 - Overview of Item class hierarchy

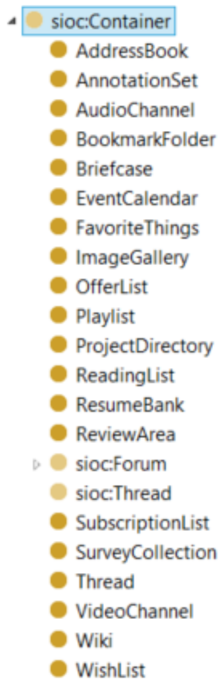


Figure 12 - Overview of Container class hierarchy

The following is an example of a Tweet represented in SocialML

```
<http://www.twitter.com/MariamButt88/status/603372152654471168>

    a         item:MicroBlogPost , item:Tweet ;

    dct:created     "2015-05-27T01:28:25Z"^^xsd:dateTime ;

    dct:creator     <http://www.twitter.com/MariamButt88> ;

    dct:identifier  "603372152654471168"^^xsd:long ;
    dct:language    "en" ;

    dct:source      <http://www.twitter.com> ;

    place:hasPlace  <http://api.twitter.com/1/geo/id/6ef29a7e78ca38a5> ;

    place:location  <geo:37.42302101,-121.92362051> ;

    item:hasLink    <http://www.theguardian.com/world/2015/may/26/french-
journalist-poses-muslim-convert-isis-anna-erelle?CMP=fb_gu> ;

    item:text       "This is really terrifying.\n\nhttp://t.co/sZY3LaUlBw"
;

    item:url
<http://www.twitter.com/MariamButt88/status/603372152654471168> .
```

The Item Ontology can be extended to accommodate specific properties for social objects. For example, Flickr provides a set of properties that refer to images at different resolutions. For this purpose a Flickr ontology extending the Item ontology has been defined to accommodate the specific properties. A Flickr Image is encoded below using SocialML:

```
<https://www.flickr.com/photos/fredarmitage/15377842444/>

    a             item:Photo ;

    dct:created         "2014-12-11T13:40:30Z"^^xsd:dateTime ;

    dct:creator         <http://www.flickr.com/people/58746120@N00> ;

    dct:dateAccepted    "2014-12-11T21:17:43Z"^^xsd:dateTime ;

    dct:description     "Found an old waterproof video camera with a 5mp camera. What better tool
to use today 12/11/2014 in the big storm?" ;

    dct:modified        "2015-04-30T08:12:08Z"^^xsd:dateTime ;

    dct:subject         "flash flood" , "wind" , "california" , "12/11/2014 storm" , "flooding" ,
"flood" , "12/11/2014" , "sewage" , "overflowing" , "overflow" , "inundations" , "inundation" ,
"storm" , "weather" , "street" , "water" , "fall" , "rainfall" , "rain" , "san francisco" , "sf"
, "drought" , "feet in water" , "rain storm" , "wind storm" , "pineapple express" , "pedestrian"
, "crossroad" , "flooded crossroad" , "san francisco storm" , "san francisco rain" , "rainy" ,
"flooded" ;

    dct:title           "Heard in the street today, I quote: \"Drought? What #drought?\"" ;
    flickr:large_1600_url   <https://farm8.static.flickr.com/7527/15377842444_3cf3dca267_h.jpg> ;

    flickr:large_2048_url   <https://farm8.static.flickr.com/7527/15377842444_3cf3dca267_k.jpg> ;

    flickr:large_square_url <https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_q.jpg> ;

    flickr:large_url        <https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_b.jpg> ;

    flickr:medium_640_url   <https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_z.jpg> ;

    flickr:medium_800_url   <https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_c.jpg> ;

    flickr:medium_url       <https://farm8.staticflickr.com/7527/15377842444_3cf3dca267.jpg> ;

    flickr:photoId          "15377842444" ;

    flickr:small_320_url    <https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_n.jpg> ;

    flickr:small_square_url <https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_s.jpg> ;

    flickr:small_url        <https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_m.jpg> ;

    place:hasLocation    [ a         place:Place ;

                  place:adminLevel1 <http://where.yahooapis.com/v1/place/2347563> ;

                  place:adminLevel2 <http://where.yahooapis.com/v1/place/12587707> ;

                  place:country     <http://where.yahooapis.com/v1/place/23424977> ;

                  place:locality    <http://where.yahooapis.com/v1/place/2487956> ;

                  foaf:based_near   [ a      wgs84:Point , point:Point ;

                            geom:asWKT "POINT(37.78455352783203 -
122.4045181274414)"^^geosparql:wktLiteral ;

                            wgs84:lat  "37.78455352783203"^^xsd:double ;

                            wgs84:long "-122.4045181274414"^^xsd:double

                          ]

                 ] ;

    item:originalURL     <https://farm8.staticflickr.com/7527/15377842444_ff31e38086_o.jpg> ;
    item:thumbnailURL    <https://farm8.staticflickr.com/7527/15377842444_3cf3dca267_t.jpg> ;

    item:url             <https://www.flickr.com/photos/fredarmitage/15377842444/> .
```

A.3.2. Social Activity Microtheories

A.3.2.1 Core Social Activity Microtheory

In the most basic sense, a social "Activity" is a semantic description of a completed or ongoing action on social objects. It is the goal of this microtheory to provide a vocabulary that is sufficient to express metadata about activities on social networks in a rich, human-friendly but machine-processable and extensible manner. In its simplest form, an `Activity` consists of an `actor`, an `action`, an `object`, and a `target`. It tells the story of a person performing an action on or with an object -- "Geraldine posted a photo to her album" or "John shared a video".

Experience with early versions of the ontology demonstrated that there is no "one size that fits all". In some cases a very simple direct representation of an activity as a verb (represented as an OWL Property) is preferred for ease of consumption. In other scenarios a more complex representation is needed to capture the nuances of an Activity. The SocialML Activity Microtheory accommodates both viewpoints.

Direct action verb

This simplest representation of a activity is to directly state that some agent (represented as a `foaf:Agent` or one of its subclasses) is performing an action on a social object. To represent specific actions that the agent perform, profiles may define sub-properties of the generic property `activity:actsOn`. SocialML provides as an example a microtheory that specialize actions verbs for online social media based on the taxonomies defined ActivityStrea.ms 1.0 standard.

N-ary Activity

Occasionally, it is advantageous to have an explicit representation of the action that the agent performs on an object. This is supported by the `activity:Action` class. The situation of an Agent fulfilling that action on an social object is then expressed through instances of the reified `activity:Activity` or one of its subclasses. This also makes it possible to annotate the activities with qualifying information such as location where the activity occurs, the time frame (duration) of the activities, the site on which the activity is performed, and so on. Figure 13 shows the model of the Activity in the ontology.

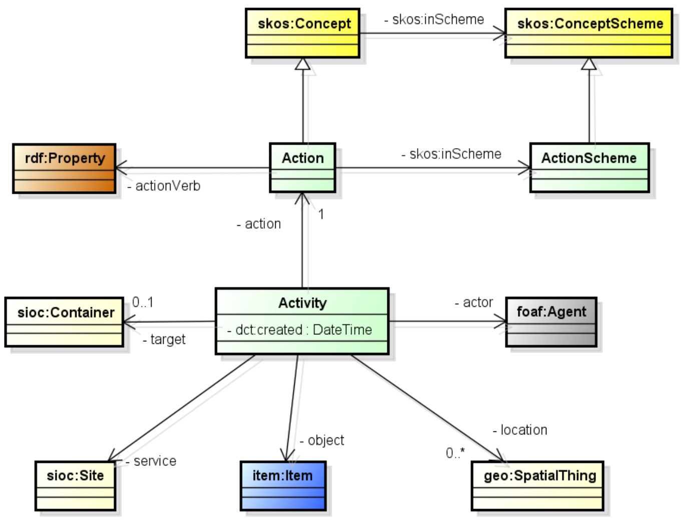


Figure 13 - Activity Model

Action definitions can be defined as a hierarchical taxonomy. For this reason, the class `activity:Action` is defined as a subclass of `skos:Concept` and we introduce the class `rel:ActionScheme` subclass of `skos:ConceptScheme` to indicate taxonomies of Actions. This approach is consistent with the ISO 19150-Part 2 standard. This may be used to create a taxonomy of actions for a given environment (social web site for example). We also define a meta-level property which is used to annotate an `activity:Action` instance with a sub-property of `activity:actionVerb` that can be used to directly indicate the action verb for ease of query. The semantics of `activity:actionVerb` can be expressed using a second closure rule:

```
CONSTRUCT {

  ?agent ?actionverb ?object.

} WHERE {

 [] a activity:Activity;

  activity:subject  ?agent;

  activity:object  ?object;

  activity:action  [ activity:actionVerb ?actionVerb ].

}
```

Tool chains generate `activity:Activity` instances and then apply this closure rule to add any corresponding short-cut specializations of `activity:actionVerb`. Our approach differs with the W3C Activity Stream model by taking a soft-typing approach to describe Actions as it provides more flexibility to plugin fast changing action taxonomies without the need to modify the core Activity ontology, but also to use SKOS mapping mechanisms to bridge Actions defined in different taxonomies (skos:exactMatch, skos:closeMatch, skos:broadMatch, skos:narrowMatch).

An example of Action definition in the realm of Social Web is shown in the listing below.

```
:Share a          activity:Action ;

    rdfs:label      "Share" ;

    activity:actionVerb :share ;

    skos:definition   "Indicates that the actor has called out the
object to readers. In most cases, the actor did not create the
object being shared, but is instead drawing attention to it." ;

    skos:inScheme    :SocialWebActionScheme .



:share rdfs:comment    "Indicates that the actor has called out the
object to readers. In most cases, the actor did not create the
object being shared, but is instead drawing attention to it." ;

    rdfs:label     "share" ;

    rdfs:subPropertyOf activity:actsOn
```

A.3.2.2 Social Web Activities Microtheory

This microtheory defines the concept of `OnlineActivity` as a subclass of `activity:Activity`. It defines activities that occur online (mostly on social web). The microtheory defines ninety-one (91) `Action` instances along with the associated action verb (as OWL Property) in a `SocialWebActionScheme` instance. The actions are drawn from the ActivityStrea.ms 1.0 standard, as an extension of Atom Format. Our semantic based approach provides an extensible framework to add more actions but also creating hierarchical action classification.

A.3.3 Social Group Microtheories

A.3.3.1 The Group Microtheory

The Group Microtheory defines a core vocabulary for social groups. The ontologies extend the Collection ontology, which itself extends the mereology (part-whole theory). This enables mereological reasoning on groups (formal organization, terrorist groups etc.). The central concept in this microtheory is `group:Group` (subclass of `col:Collection`**).** This class is equivalent to `foaf:Group`. The ontology introduces predicates to capture subgroup relationships (`group:subGroupOf` and `group:hasSubGroup`) and group membership relationship (`group:groupMember`). This ontology is designed to be extended to model various kinds of groups (e.g. community, terrorist, religious, ethnic groups), or formal groups such as corporations, governments, or military organizations.

A.3.3.2. The Organization Ontology

This organization ontology extends the Group Microtheory and is designed to enable publication of information on organizations and organizational structures including governmental organizations. We aligned this ontology and the membership relationships with the W3C Organization Ontology. It is designed to provide a generic, reusable core ontology to be extended or specialized for use in particular situations. The ontology provides terms to support the representation of the following:

- organizational structure
  - notion of an organization
  - decomposition into sub-organizations and units
  - purpose and classification of organizations
- reporting structure
  - membership and reporting structure within an organization
  - roles, posts, and the relationship between people and organizations
- location information
  - sites or buildings
  - locations within sites
- organizational history (merger, renaming, re-purposing)

The ontology does not provide category structures for organization type, organization purpose or roles. Because different domains will have different requirements for classification of such concepts, the ontology provides only the core base concepts needed to allow extensions to add specific subclass structures or classification schemes as required. This follows the minimal ontological commitment strategy. Users of the ontology are encouraged to define profiles to enhance interoperability by specifying particular controlled vocabularies to use for these concepts.

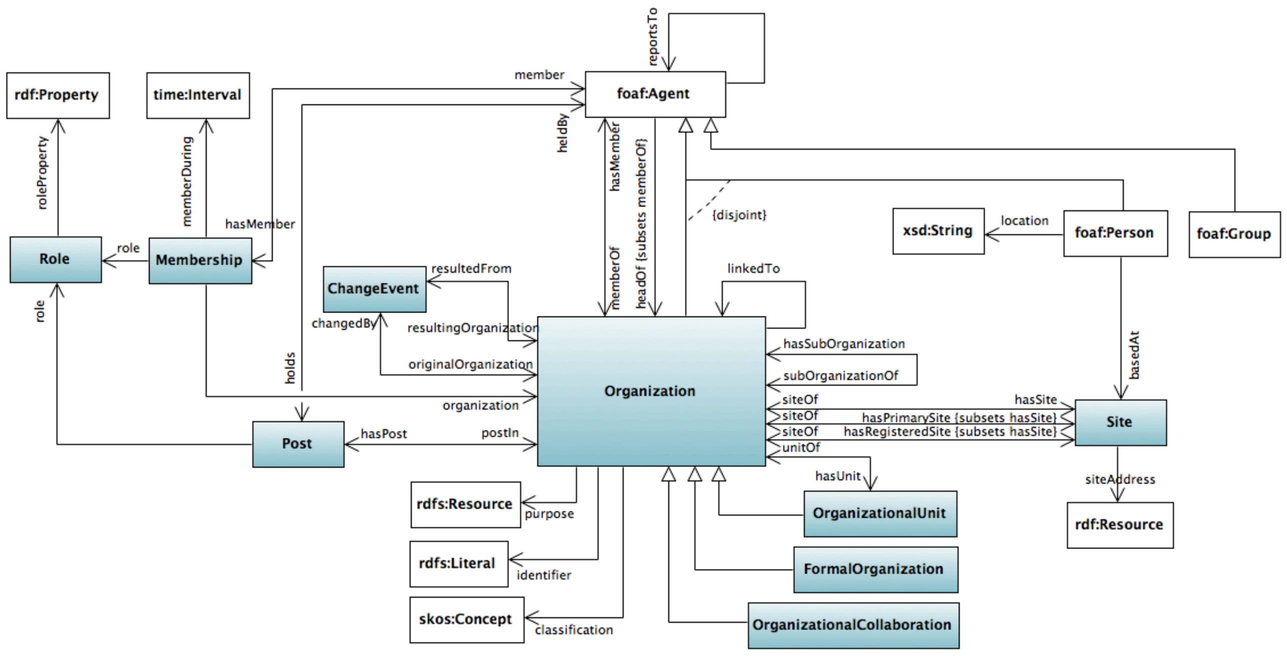


Figure 14 - Organization Ontology

The core class in the ontology is `org:Organization` is intended to be applicable to a broad range of organizations. It is equivalent to W3C `w3c-org:Organization` and `foaf:Organization`, and is a subclass of `group:Group` concept. It represents a collection of people organized together into a community or other social, commercial or political structure. The group has some common purpose or reason for existence which goes beyond the set of people belonging to it. An organization may itself be able to act as an agent.

We distinguish a particular sub-class of organization `org:FormalOrganization` to indicate organizations that are recognized in the world at large, in particular in legal jurisdictions, with associated rights and responsibilities. Examples include corporations, charities, governments or religious organizations.

The ontology supports the notion of organizations being composed of other organizations in some hierarchy. The relations `org:subOrganizationOf` and `org:hasSubOrganization` establish these hierarchical links and are sub-properties of `group:subGroupOf` and `group:hasSubGroup` respectively. The object property `org:hasSubOrganization` represents hierarchical containment of Organizations or OrganizationalUnits. `org:subOrganizationOf` indicates an organization which is a subpart or child of another organization.

In some cases the sub-organization can be regarded as standalone - for example a legally recognized business may be part of a larger group or holding company. In other cases it is useful to refer to departments or organizational units such as the IT department which only have meaning within the Context of the containing organization. The ontology supports that situation through a specialization of `org:Organization` called `org:OrganizationalUnit` The class `org:OrganizationalUnit` is defined as an organization such as a department or support unit which is part of some larger `org:Organization` and only has full recognition within the context of that `org:Organization`. In particular that unit would not be regarded as a legal entity in its own right. Units can be large and complex containing other units. An alternative name that the general public also uses is Department. For convenience it also provides the relations `org:hasUnit` and `org:unitOf` which are specializations of the generic sub-organization links.

A.3.2.4. Person Microtheories

The Person microtheories capture profile information of a person or persona (digital representation of a Person online). A person can have many personas online. They cover the following aspects:

Personal identification: name, first name, alias, citizenship, address, etc.

Physical characteristics: scars, hair color, etc.

Cognitive characteristics: belief, interest, expertise, skills, education, strength, weak nesses, etc.

A.3.2.4.1 Personal Identification

The main concept in our Person ontology is Person. It is an equivalent class to `foaf:Person`, this means that most of properties of FOAF can be used to describe a person. Any property which may be used to verify aspects of a person's personal identity is considered a personal identification entity. Many of these properties are derived from the FOAF ontology, which includes `foaf:firstName`, `foaf:familyName`, `foaf:name`, and `foaf:givenName`. In many instances, persons can also be identified by some aliases. We introduce in our ontology the property: alias to capture different aliases a person can have. The ontology also leverages the Identifier ontology to define different type of `PersonalIdentifier`, as illustrated in Figure 15. The Identifier are captured as classes on which additional properties can be added to qualify further the identification name. For example, an alias may have a reliability score to indicate how reliable is the information about the identification.

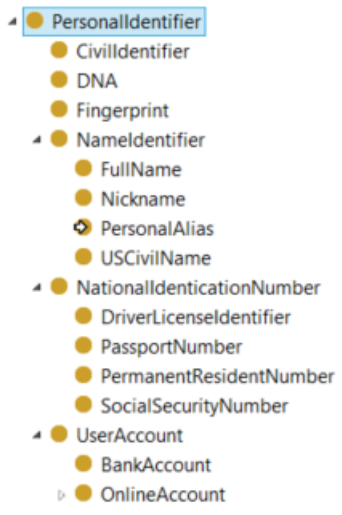

```
PersonalIdentifier
    CivilIdentifier
    DNA
    Fingerprint
    NameIdentifier
        FullName
        Nickname
        PersonalAlias
        USCivilName
    NationalIdenticationNumber
        DriverLicenseIdentifier
        PassportNumber
        PermanentResidentNumber
        SocialSecurityNumber
    UserAccount
        BankAccount
        OnlineAccount
```

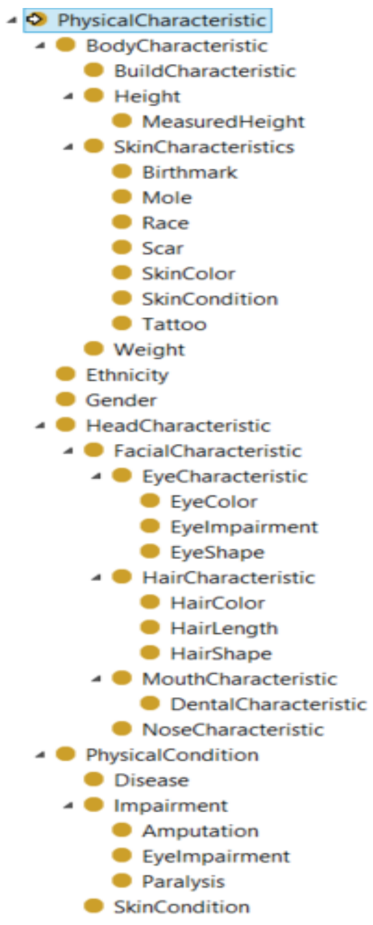
Figure 15 - Personal Identifier class hierarchy

The following example show how personal identifications are captured:

```
:Anwar_Al-Awlaki

 a person:Person;

 foaf:family_name 'al-Awlaki';

 foaf:firstName 'Anwar';

 person:alias 'Anwar al-Aulaqi',

        'Anwar Nasser Abdulla Aulaqi'

        'Anwar Aulaqi'.
```

A.3.2.4.2 Physical Characteristics

This microtheory captures properties related to physical characteristics of a person. Examples of physical characteristics include `Gender`, `HairColor`, `EyeColor`, `FacialFeature`, `Impairement` with associated properties `gender`, `hairColor`, `eyeColor`, `facialFeature`, `impairement`. They are subclasses of `PhysicalCharacteristics` and most of them are also subclasses of `skos:Concept`. Using this approach, it is possible to build taxonomies for `PhysicalCharacteristic` types that accommodates specific domain (immigration, law enforcement) and perform inferencing and classifications of physical characteristics. In case a taxonomy can be defined (for describing `Birthmarks` for example), the `PhysicalCharacteristic` class provides a property `description` to capture a textual description of the characteristic and well as `foaf:depiction` to attach visual representation. Figure 16 shows an overview of the class hierarchy of the Physical CharacteristicsFigure 16.



```
PhysicalCharacteristic
  BodyCharacteristic
    BuildCharacteristic
    Height
      MeasuredHeight
    SkinCharacteristics
      Birthmark
      Mole
      Race
      Scar
      SkinColor
      SkinCondition
      Tattoo
    Weight
  Ethnicity
  Gender
  HeadCharacteristic
    FacialCharacteristic
      EyeCharacteristic
        EyeColor
        EyeImpairment
        EyeShape
      HairCharacteristic
        HairColor
        HairLength
        HairShape
      MouthCharacteristic
        DentalCharacteristic
      NoseCharacteristic
  PhysicalCondition
    Disease
    Impairment
      Amputation
      EyeImpairment
      Paralysis
    SkinCondition
```

Figure 16 - Physical Characteristics Class Hierarchy

The following example shows how physical characteristics are used on a Person instance using physical characteristics taxonomies.

```
:Anwar_Al-Awlaki

  a person:Person;

  phys:hairColor phys-taxo:BlackHairColor;

  phys:eyeColor  phys-taxo:HazelEyeColor;

  phys:facialFeature phys-taxo:Beard.
```

A.3.2.4.3 Cognitive Characteristics

This microtheory captures properties related to cognitive characteristics of a person. Examples of cognitive characteristics include `Interest`, `Belief`, `Competence`, `Education`, and `Preference` with associated properties `interest`, `belief`, `competence`, `education`, `preference`. They are subclasses of `CognitiveCharacteristics` and most of them are also subclasses of `skos:Concept`. Using this approach, it is possible to build taxonomies for `CognitiveCharacteristic` types that accommodate a specific domain (immigration, law enforcement) and perform inferencing and classifications of physical characteristics. In case a taxonomy can be defined (for describing `Interest` for example), the `CognitiveCharacteristic` class provides a property `description` to capture a textual description of the characteristic. Figure 16 shows an overview of the class hierarchy of the Cognitive Characteristics.

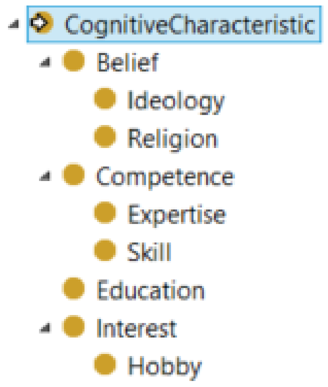


Figure 17 - Cognitive Characteristics

The following example shows how cognitive characteristics are used on a Person instance using cognitive characteristics taxonomies

```
:Anwar_Al-Awlaki

  a person:Person;

  coc:skill :Preacher;

  coc:belief  religion:Shiite;

  coc:expertise :Imam .
```

A.3.2.5 Social Relationships Microtheories

The Social Relationship microtheories provide an ontological model to describe social relationships between two agents (person or organization). The relationships are mostly captured as OWL Object Properties, but can be reified as a Relationship object to add additional information such as the closeness (strength) of a relationship, the time period in which a relationship was valid. The microtheories are modularized according the type of the source and target of the relationships.

- *Core Relationships*: defines core concepts of n-ary `rel:Relationship`, `rel:Role` and generic base property `rel:linkedTo`.
- *Interpersonal Relationships*: describes relationships between two persons
- *Inter-organizational Relationships*: describes relationships between two organizations
- *Membership Relationships*: describes relations between person and organizations.

We argue that ontological representations of social networks such as FOAF would need to be extended with a framework for modeling and characterizing social relationships for two principle reasons: (1) to support the automated integration of social information on a semantical basis and (2) to capture established concepts in Social Network Analysis, which provides the most significant toolkit for processing social networks with the purpose of understanding social structure and its effects.

These ontologies are very useful to perform semantic social network analysis when relationships between agents can be inferred using OWL semantics. For example, it is trivial to extract from a complex social network, family relationships networks versus working relationships network. This is commonly used techniques used in counter-terrorism or financial fraud analysis.

A.3.2.5.1 Core Relation Microtheory

This lightweight microtheory provides a number of ways to represent the relationship between agents (Person, Group, and Organization). Experience with early versions of the ontology demonstrated that there is no "one size that fits all". In some cases a very simple direct representation of a relationship is preferred for ease of consumption. In other cases a more complex representation is needed to capture the nuances of a relationship. The SocialML Core Relationship Microtheory accommodates both viewpoints.

Direct relation

This simplest representation of a relationship is to directly state that some agent (represented as a `foaf:Agent` or one of its subclasses) is `rel:linkedTo` another agent. To represent specific roles that the agent plays, profiles may define sub-properties of `rel:linkedTo`. SocialML provides three microtheories that specialize relationship roles according three categories: interpersonal, inter-organizational and membership relationships. These microtheories will be described more in details below.

N-ary relationship

Sometimes, it is advantageous to have an explicit representation of the relationship role that the agent fulfils (e.g. for publication of responsibilities associated with the role). This is supported by the `rel:Role` class. The situation of an Agent fulfilling that role with another agent (an organization for example) is then expressed through instances of the `rel:Relationship` n-ary relationship or one of its subclasses. This also makes it possible to annotate the relationship with qualifying information such as duration, strength of a relationship (closeness), salary, reference to the employment contract and so forth. Figure 18 summarizes the model for Relationship entity.

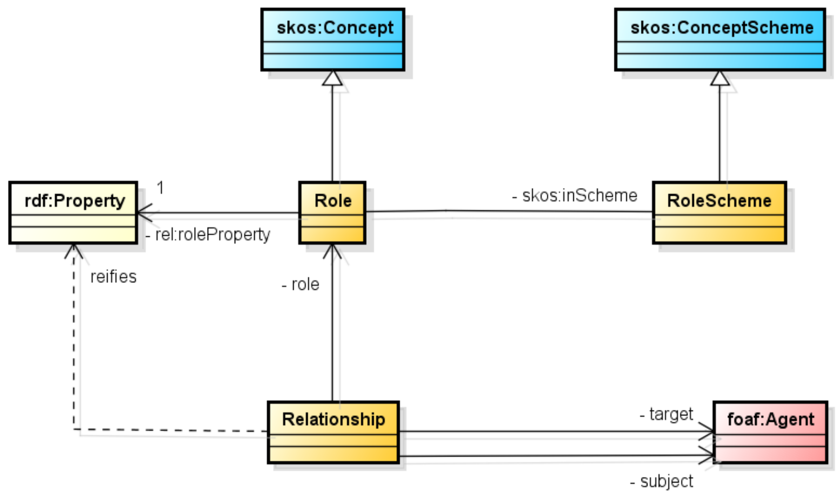


Figure 18 - Relationship Model

Role definitions can be defined as a hierarchical taxonomy. For this reason, the class `rel:Role` is defined as a subclass of `skos:Concept` and we introduce the class `rel:RoleScheme` subclass of `skos:ConceptScheme` to indicate taxonomies of roles. This approach is consistent with the ISO 19150-Part 2 standard. This could be used to create taxonomy of roles within an organization. We also define a meta-level property which is used to annotate a `rel:Role` instance with a sub-property of `rel:linkedTo` that can be used to directly indicate the role for ease of query. The semantics of `rel:roleProperty` can be expressed using a second closure rule:

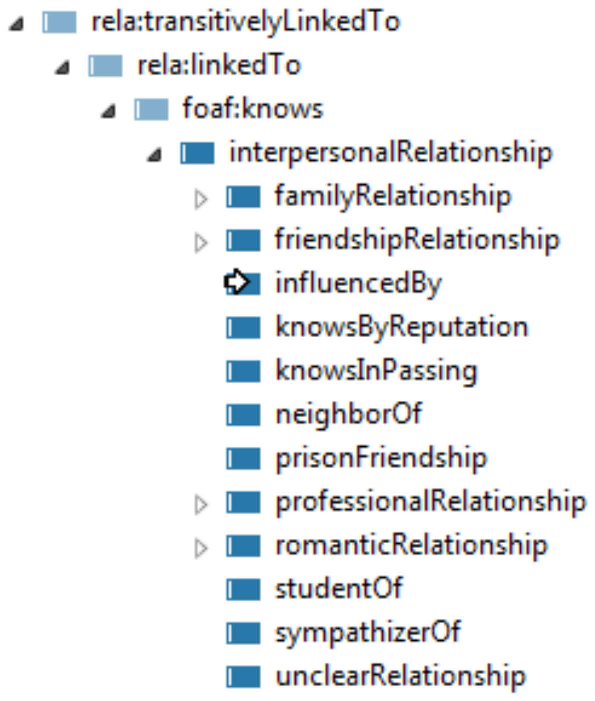
```
CONSTRUCT {

  ?agent1  ?roleprop ?agent2.

} WHERE {

 [] a rel:Relationship;

  rel:source  ?agent1;

  rel:target  ?agent2;

  rel:role     [ rel:roleProperty ?roleprop ].

}
```

Tool chains may generate `rel:Relationship` instances and then apply this closure rule to add any corresponding short-cut specializations of `rel:linkedTo`.

A.3.2.5.2 Interpersonal Relationships

An interpersonal relationship is an association between two persons or personas. The concept of inter- personal relationships involves social associations, connections, or affiliations between two or more people. This association may be based on inference, regular business interactions, or some other type of social commitment.

The interpersonal relationships are divided into four specialized groups: familial relationships (ex. :`sonOf`), friendships (ex. :`friendOf`), romantic relationships(ex. :`wifeOf`), and professional relationships (ex. :`colleagueOf`). Figure 19 shows a partial hierarchy of how interpersonal relationships are structured in the ontology.



```
rela:transitivelyLinkedTo
    rela:linkedTo
        foaf:knows
            interpersonalRelationship
                familyRelationship
                friendshipRelationship
                influencedBy
                knowsByReputation
                knowsInPassing
                neighborOf
                prisonFriendship
                professionalRelationship
                romanticRelationship
                studentOf
                sympathizerOf
                unclearRelationship
```

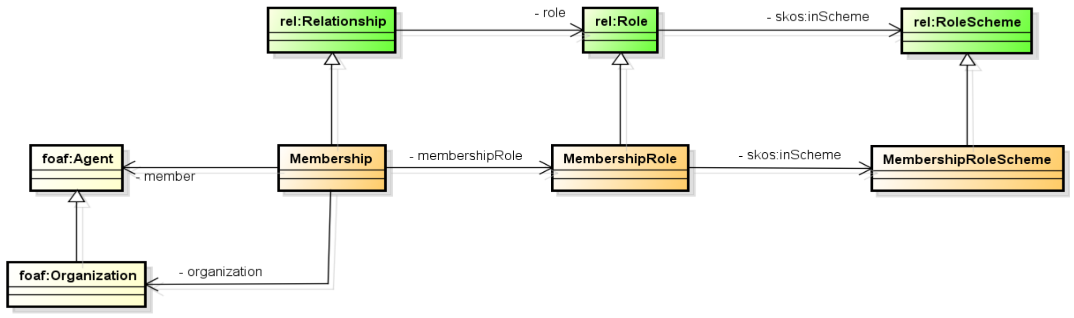
Figure 19 - Interpersonal Relationships Hierarchy

A.3.2.5.2 Inter-organizational Relationships Ontology

An inter-organizational relationship is defined when an organization is linked to another organization. There are three types of organizational affiliations: Membership Relationships, Structural Relationships, and Organizational Relationships. A membership relationship refers to a link between a person and an organization. A structural relationship refers to links between organizations, where there is no significant relationship. An organizational relationship refers to links between organizations where there is a thorough relationship.

A.3.2.5.3 Membership Microtheory

The Membership microtheory represents relationships between an Agent, an Organization and a Role. It is possible to directly indicate membership, independent of the specific Role, through use of the `org:memberOf` property or one its sub-properties. The ontology extends the core Relationship ontology by introducing a subclass of `rel:Relationship` called `mbrship:Membership`.



Revision history

Date	Release	Editor	Primary clauses modified	Description
20.02.2015	0.0.1	MRI	All	Initial document
10.06.2015	0.0.2	SJI	All	First consolidated version
11.06.2015	0.0.3	MRI	5.2, 5.3, 5.4, 5.5, 5.6	Update of contributions from other partners
11.06.2015	0.0.4	SJI	6, 7, 8	Enhancement of the sections on accomplishments, lessons learned and recommendations
12.06.2015	0.0.5	MRI	All	Edits on consolidated document
14.06.2015	0.0.6	SJI	All	Review of content
23.06.2015	0.0.7	STF	All	Integration of Linked Data approach
26.06.2015	0.0.8	SJI	All	Several edits and integration of further contributions from partners.
15.07.2015	1.0.0	SJI	All	Final edits
03.09.2015		Carl Reed	Various	Prepare for publication