Terra Cognita

and

Semantic Sensor Networks

In Conjunction with the 13th International Semantic Web Conference (ISWC 2014)

Riva del Garda - Trentino, Italy
October 20, 2014
Workshop Proceedings

Contents

Terra Cognita 2014

6th International Workshop on the Foundations, Technologies and Applications of the Geospatial Web

Riva del Garda - Trentino, Italy, 20 October 2014

Introduction

The wide availability of technologies such as GPS, map services and social networks, has resulted in the proliferation of geospatial data on the Web. Similarly, the amount of geospatial data extracted from the Web and published as Linked Data is increasing. Together with the dissemination of Web-enabled mobile devices these continually growing data have given rise to a number of innovative services and applications. With the location of users being made available widely, new issues such as those pertaining to security and privacy arise. Emergency response, context sensitive user applications, and complex GIS tasks all lend themselves toward solutions that combine both the Geospatial Web and the Semantic Web.

The workshop will bring together researchers and practitioners from various disciplines, as well as interested parties from industry and government, to advance the frontiers of this emerging research area. Bringing together Semantic Web and geospatial researchers helps encourage the use of semantics in geospatial applications and the use of spatial elements in semantic research and applications. The field continues to gain popularity, resulting in a need for a forum to discuss relevant issues.

Organization

Terra Cognita 2014 was organized in conjunction with the International Semantic Web Conference 2014 at Riva del Garda, Italy.

Program Committee Chairs

Kostis Kyzirakos (Centrum Wiskunde & Informatica - CWI, Amsterdam, The Netherlands)
Rolf Gruetter (Swiss Federal Research Institute WSL, Birmensdorf, Switzerland)
Dave Kolas (Raytheon BBN Technologies, Columbia, MD, USA)
Matthew Perry (Oracle Corp., Nashua, NH, USA)

Program Committee

Alia Abdelmoty (Cardiff University, UK)
Spiros Athanasiou (IMIS/RC Athena, Greece)
Sotiris Batsakis (University of Huddersfield, UK)
Jon Blower (University of Reading, UK)
Oscar Corcho (Universidad Politécnica de Madrid, Spain)
Isabel Cruz (University of Illinois at Chicago, IL, USA)
Mike Dean (Raytheon BBN Technologies, Ann Arbor, MI, USA)
Curdin Derungs (University of Zürich, Switzerland)
John Goodwin (Ordnance Survey, UK)
Glen Hart (Ordnance Survey, UK)
Andreas Harth (AIFB, Karlsruhe Institute of Technology, Germany)
Krzysztof Janowicz (University of California, Santa Barbara, CA, USA)
Marinos Kavouras (National Technical University of Athens, Greece)
Manolis Koubarakis (National and Kapodistrian University of Athens, Greece)
Stefan Manegold (Centrum Wiskunde & Informatica - CWI, Amsterdam, the Netherlands)
Axel-Cyrille Ngonga Ngomo (University of Leipzig, Germany)
Alexandros Ntoulas (Microsoft Research, Mountain View, CA, USA)
Özgür Lütfü Özcep (Institute for Softwaresystems, Hamburg University of Technology, Germany)

Dieter Pfoser (George Mason University, Fairfax, VA, USA)
Clemens Portele (Interactive instruments, Bonn, Germany)
Ross Purves (University of Zürich-Irchel, Switzerland)
Thorsten Reitz (Fraunhofer Institute for Computer Graphics Research, Darmstadt, Germany)
Thomas Scharrenbach (Universtiy of Zurich, Switzerland)
Timos Sellis (RMIT University, Melbourne, Australia)
Spiros Skiadopoulos (University of Peloponnese, Tripoli, Greece)
Kerry Taylor (CSIRO & Australian National University, Canberra, Australia)
Raphael Troncy (EURECOM: School of Engineering & Research Center, Biot, France)
Nancy Wiegand (University of Wisconsin, Madison, WI, USA)
Stefan Woelfl (Department of Computer Science, University of Freiburg, Germany)

Publishing Reference Geodata on the Web: Opportunities and Challenges for IGN France

Ghislain A. Atemezing[1], Nathalie Abadie[2],
Raphaël Troncy[1], Bénédicte Bucher[2]

[1] EURECOM, SophiaTech Campus, France,
[2] Université Paris-Est, IGN /SRIG, COGIT, Saint-Mandé

Abstract. The French national mapping agency (IGN) produces several different but complementary geographic vector reference databases delivered in traditional GIS formats. However, linked data users have different expectations and habits, such as the need to browse an entire data catalogue in RDF using the "follow-your-nose" navigation capacity from one graph to another. Besides, traditional GIS data formats are not interoperable with RDF. Yet, all these geographic datasets could be used with benefits on the Web of data, either with direct georeferencing through geographic primitives, or indirect one through postal addresses. In this paper, we aim to contribute to the georeferencing of datasets published on the Web of data by providing such resources for French context. Firstly, we propose two vocabularies designed for representing structured geometries defined with coordinates expressed in any Coordinates Reference System (CRS). Secondly, we reuse these vocabularies and the CRSs' dataset to publish a reference dataset on administrative units that can also be reused for indirect georeferencing purposes. Finally, we also propose two vocabularies for describing geographic feature types. In addition to these resources, we also present a comprehensive workflow for easily publishing geographic data on the Web of data.

Keywords: Ontology Design, Geospatial Data, Linked Data, Georeferencing, Structured Geometry, Coordinate Reference System, data.ign.fr

1 Introduction

The French national mapping agency (IGN) produces several different but complementary geographic vector reference databases (BD TOPO®, BD CARTO®, BD ADRESSE®, etc.). They are structured according to object-oriented application schemas (ISO 19109). As an example, GEOFLA® database contains data on the French administrative units. Their boundaries are described by geometries of type MultiPolygon and their properties such as toponyms, population, legal codes and hierarchical relationships are stored by attributes. All these databases are provided in traditional GIS formats (ESRI shapefiles or GML). As required by the INSPIRE Directive, IGN provides users with a data visualization portal[3].

[3] `http://www.geoportail.gouv.fr/accueil`

However, linked data users have different expectations and habits. They need to browse the entire data catalogue in RDF and wish to have *"follow-your-nose"* navigation possibility from one graph to another. Besides, GIS data formats are not interoperable with RDF. Indeed, many resources published on the Web of data are georeferenced, either directly through geographic coordinates, geometric primitives or indirectly through postal addresses, names of administrative units or points of interest. According to LOD cloud statistics, the properties `geo:long` and `geo:lat` of the W3C vocabulary Geo[4] are respectively used in 530 450 and 530 515 triples within 59 datasets, while 36 datasets reuse classes defined by 6 different vocabularies describing postal addresses[5]. We have also identified more than 80 properties with semantic meaning closed to :locatedIn or :hasLocation.

In this article, we propose to contribute to the georeferencing of datasets published on the Web of data by providing some useful resources. Firstly, we propose two vocabularies designed for representing structured geometries defined with coordinates expressed in any Coordinates Reference System (CRS). A dataset dedicated to the description of CRSs defined and maintained by IGN France is also published and can be reused for direct georeferencing purposes. Secondly, we reuse these vocabularies and the CRSs' dataset to publish a reference dataset on administrative units that can also be reused for indirect georeferencing purposes. Finally, we also propose two vocabularies for describing geographic feature types. In addition to these resources, we also present a comprehensive workflow for easily publishing geographic data on the Web of data.

The remainder of this article is structured as follows. In Section 2, we present some technical considerations on data georeferencing and publishing on the Web of data. The Section 3 describes the vocabularies developed for topographic features and their geometries. We then present the generation and publication of administrative units (Section 4) and the French gazetteer in Section 5. We conclude the paper with some challenges (Section 6). Finally some conclusions are drawn.

2 Georeferencing data and Technical considerations

Georeferencing data either by direct or indirect spatial reference requires some reference datasets that can be used as the spatial frame for anchoring these thematic data. Especially, it requires data on both CRSs and named places, which must be published on the Web of data.

2.1 Direct georeferencing of data on the Web

Modeling direct location information such as coordinates or vector data geometries in RDF still poses some challenges. In [1], we have conducted a survey of the vocabularies used for representing geographical features from vocabularies

[4] `http://www.w3.org/2003/01/geo/wgs84_pos#`
[5] `http://stats.lod2.eu/`

of feature types to vocabularies for geometric primitives which provide ways for representing extents, shapes and boundaries of those features. Most of vocabularies dedicated to geometry representation reuse W3C Geo vocabulary which allows only WGS84 coordinates, such as NeoGeo[6]. With the rise of the Open Data movement, more and more publishers including governments and local authorities are releasing legacy data that are georeferenced using others CRSs. For example, IGN France releases data using different projected CRSs depending on the geographic extent of each dataset. In order to overcome this limitation on CRSs, the vocabulary designed by OGC GeoSPARQL standard does not reuse W3C Geo vocabulary but proposes another class "Point" instead. Geometries of geographical data represented in RDF with the GeoSPARQL vocabulary are represented by literals encoded consistently with other OGC standards. `gsp:wktLiteral` and `gsp:gmlLiteral` are thus respectively derived from Well-Known Text and GML encoding rules. In `wktLiteral` and `gmlLiteral`, the CRS used to define the coordinates of the point is identified by a dereferenceable URI which is explicitly stated at the beginning of the literal. This way of associating coordinate reference systems with geometries has the advantage of being consistent with Linked Data principles: each CRS is identified with a dereferenceable URI. The main drawback is that such literals cannot be easily queried with SPARQL, unless using regular expression-based filters. To overcome this limitation, we propose in the geometry vocabulary presented in Section 3 to associate each geometry to the CRS used by its coordinates with the property `geom:crs`.

2.2 Indirect georeferencing of data on the Web

Modeling indirect location information such as administrative units or named points of interest in RDF is preferably done by identifying such geographic features with URIs and describing them by their properties, so that they can be referenced by other datasets. This is the case in one of the most reused datasets of the Web of data, namely Geonames[7]. However, there are yet very few reference datasets for the French territory on the Web of data. A simple example is the current resource for *Paris* in the French DBpedia[8]. The department's name associated to this resource is a literal named "Paris" and the different arrondissements composing the city are modeled as `skos:Concept` instead of `dbpedia-owl:Place`. Even Geonames data remain very limited, as French administrative units are provided as simple geometries (POINT). The "Official Geographic Code"[9] published by the French Statistical Institute (INSEE) is the most up-to-date and accurate dataset on French administrative units, but unfortunately it contains no geometrical description of their boundaries. This has the consequence of not having a baseline during mapping process for application developers trying to consume specific data coming from France. Datasets

[6] `http://geovocab.org/doc/neogeo/`
[7] `http://sws.geonames.org/`
[8] `http://fr.dbpedia.org/resource/Paris`
[9] `http://rdf.insee.fr/sparql`

describing administrative units, points of interest or postal addresses with their labels and geometries, and identifying these features with URIs could be used with benefits not only for georeferencing other datasets, but also for interlinking datasets georeferenced by direct and indirect location information.

2.3 Publishing French geographic data on the Web

In order to be published on the Web of data, geographic data must be transformed from their traditional GIS formats into RDF. They must be refined using suitable vocabularies which can be either created for that specific purpose or reused thanks to some catalogue such as LOV[10] [9]. Geographic features must be identified by URIs created according to well-defined policies. Licenses must be attached to the datasets. Additionally, data must be interlinked with various datasets already published on the Linked Open Data cloud. All these steps require specific tools and skills, so that only a few geographic datasets have been published yet in RDF by National Mapping Agencies.

The Ordnance Survey Linked Data Platform[11] has published three products as Linked Data : Gazetteer, Code-Point and the administrative geography for Great Britain [3]. They also provide a wide range of APIs for accessing the different datasets. For visualizing, a Linked Data API[12] is used on top of the datasets. Similar initiative was presented in [2] for Spanish geographical datasets. Although the authors use an ontology network for the modeling, it is difficult at the moment to reuse their vocabulary for geometry because it is more specific to their use-case. However, the availability of complex geometry both in OGC standards and in more-structured RDF is interesting and should be adopted for our use case. Regarding tools integrating workflow for dealing with geodata, the GeoKnow stack[13] offers a set of tools to publish and visualize geodata. But GeoKnow stack is more oriented to expert users in Semantic Technologies. That is why we chose the Datalift Platform [8] among other solutions because it includes almost all of the aforementioned functionalities to publish geographic data as Linked Data, and integrates a geographic data converter. Moreover, it can be used with a variety of triple stores, and more important, it is target at lay users.

3 Vocabularies for Geometries and Feature Types

Direct georeferencing of data implies representing coordinates or geometries and associating them to a CRS. This requires vocabularies for geometries and CRSs. Besides, indirect georeferencing of data implies associating them to other data on named places. Preferably, these data on named places should be also georeferenced by coordinates in order to serve as basis for data linking between indirectly

[10] `http://lov.okfn.org/dataset/lov/`
[11] `http://data.ordnancesurvey.co.uk`
[12] `http://code.google.com/p/elda/`
[13] `http://stack.linkeddata.org/download/`

and directly georeferenced datasets. In this section, we present the vocabularies that we have defined and reused for geographic data publishing.

3.1 A vocabulary for geometries

In [1], we already surveyed numerous vocabularies for representing geographical features and their geometries, either using a literal (e.g. wktLiteral) or a structured representation à la NeoGeo. We concluded the survey with some recommendations for geometry descriptions:

- the distinction of geometry versus feature and a property linking both classes (e.g. for attaching provenance information on how some points of a geometry have been collected),
- the ability to represent structured geometries (e.g. for performing simple spatial queries on the data, even when they are stored in a triple store that do not implement the GeoSPARQL standard),
- the integration of any coordinate reference system (e.g. for allowing projected coordinates for cartographic purposes).

In addition to these recommendations, we also think that the domain of the property used to link a feature to its geometry should be left empty in order to accept links between any type of resource and a geometry. This would be useful for example, to associate a person to the coordinates of their birthplace.

Extending GeoSPARQL vocabulary In order to fulfill these recommendations, we have developed a new vocabulary that re-uses and extends the existing vocabularies for representing geometries, namely:

- `http://www.opengis.net/ont/geosparql#` (prefix `gsp`). This vocabulary provides the basic concepts to represent geographical data such as SpatialObject, Feature or `Geometry`. A Feature is linked to a Geometry via the relation `gsp:hasGeometry`. The geometries are typed strings (`gsp:gmlLiteral` or `gsp:wktLiteral` corresponding respectively to the properties `gsp:asGML` and `gsp:asWKT`). The vocabulary contains also spatial functions.
- `http://www.opengis.net/ont/sf#` (prefix `sf`): This vocabulary is based on the OGC standard Simple Features Access [5]. The class `sf:Geometry` is a subclass of `gsp:Geometry`.

Reusing and extending GeoSPARQL Simple Features vocabulary with structured geometries à la NeoGeo enables us to represent geometries both with GeoSPARQL compliant literals and with structured geometries that can be handled easily with SPARQL. The extension for structured geometries consists in defining a subclass for each class from the `sf` vocabulary, and defining properties to associate its instances with a CRS and coordinates or other suitable geometric primitives. For example, the class `geom:Point` is a subclass of `sf:Point`. An instance of `geom:Point` is associated with exactly one instance of `ignf:CRS` via the property `geom:crs`, and it has exactly one coordinate X and exactly one coordinate Y. It can also have a Z coordinate. The coordinates are `xsd:double` and

correspond to the properties `geom:coordX:`, `geom:coordY:` and `geom:coordZ:` respectively. Other complex geometries are also defined, such as Linestrings, LinearRings, Polygons or MultiPolygons. Their definitions are based on the class `geom:Point`. As an example, an instance of `geom:Linestring` is defined as an instance of `geom:PointsList` which is an ordered `rdf:List` of instances of `geom:Point` designated by the property `geom:points`.

We have also defined a property `geom:geometry` with an empty domain. Thus, our proposal defines a more generic class for a POINT with the benefit of choosing the CRS of the underlying data. Figure 1 gives an overview of the relationships between the high level concepts with geometries, CRS and topographic features.

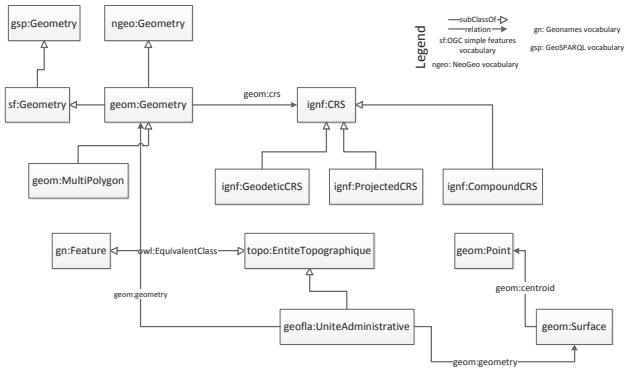


Fig. 1: High level classes of ignf, geom and topo vocabularies; relationships between them and mappings with external vocabularies.

3.2 CRS requirements for the French territory

As explained in Section 2, making explicit the CRS used in a given dataset is a very important issue when dealing with direct location data. This is especially important in the field of geographical information where different CRSs are commonly used due to technical or legal requirements. For INSPIRE Directive, CRS are considered as reference data used for linking thematic data [4], and must be described according to ISO 19111 standard. To be consistent with Linked Data principles, CRS should be identified by URIs, like in OGC proposal. Moreover, as Linked Data users are not always familiar with CRS identifiers commonly used within the geographic information community, URI used to identify CRS should use more intuitive names. Finally, consistently with our goal of contributing to a better georeferencing of data on the French territory, we need an access to the

descriptions of all French CRSs, including some deprecated but still used CRSs like "Lambert 1".

3.3 Identifying and describing CRSs on the Web

In order to fulfill the need for CRS identification and description on the Web, OGC maintains a set of URIs for identifying the most commonly used CRS. While very useful, the main disadvantage of this proposal is that the URIs defined by OGC are not very intuitive for users who are not familiar with Spatial Reference System Identifiers defined by geographic information authorities like OGC or EPSG, such as "4326" (which actually refers to a WGS84 CRS defined by the EPSG). Moreover, many CRS commonly used locally, such as deprecated French projected CRS, are not available in that registry. In addition to OGC proposal, several registries have been proposed by the geographic information community for cataloguing existing CRSs. The EPSG Geodetic Parameter Registry[14] allows querying the Geodetic Parameter Dataset gathered by the EPSG. CRSs can be retrieved by name, by code, by type or by coverage area, and their characteristics are displayed on a HTML form. Unfortunately, there is no direct access to these data through dereferenceable URIs. The Information and Service System for European Coordinate Reference Systems[15] provides an access to ISO 19111 standard-based descriptions of the main European CRSs but has the same limitation as the EPSG registry: access to the descriptions is not allowed by URI, but only through a cartographic interface. `SpatialReference.org` initiative aims at allowing users to use URI-based references to spatial reference systems, including some CRSs defined and maintained by IGN France. Besides, the proposed URL policy is not very intuitive. As an example, this URL identifies the projected system defined by IGN France, Lambert 93: `http://spatialreference.org/ref/sr-org/7527/`. Moreover, the definitions of some deprecated CRSs such as Lambert zone projected CRSs (which are still used in some datasets) seem to be referenced only for the authority EPSG and not for IGNF, which also maintains a registry of CRSs. ISO 19111 standard-based definitions of all CRSs defined and maintained by IGN France are published in an XML file[16]. References to equivalent definitions provided by the EPSG registry are explicitly stated with EPSG SRID. CRSs are identified by URIs using short names instead of numeric codes. For example, `http://registre.ign.fr/ign/IGNF/crs/NTFLAMB2E` is the URI designed for the "Lambert 2 étendu" projected system. Indeed "NTFLAMB2E"

Prefix	URI
geofla	http://data.ign.fr/def/geofla#
geom	http://data.ign.fr/def/geometrie#
ignf	http://data.ign.fr/def/ignf#
rgeofla	http://data.ign.fr/id/geofla/
topo	http://data.ign.fr/def/topo#
rtopo	http://data.ign.fr/id/topo/

Table 1: URI schemes and conventions used for vocabularies and resources .

[14] `http://www.epsg-registry.org/`
[15] `http://www.crs-geo.eu`
[16] `http://librairies.ign.fr/geoportail/resources/IGNF.xml`

is used to identify the projected system "Lambert 2 étendu" which is based on NTF (New French Triangulation) geodetic reference system. Unfortunately, this registry is still in evolution and its URIs are not dereferenceable yet.

As no existing registry fulfilled all our requirements, we have developed a vocabulary[17], inspired from the ISO 19111 schema for CRSs description. Then we have converted IGNF CRSs registry into RDF, and published this dataset on the Web with the Datalift platform[18]. Therefore, the description of the "NTF Lambert 2 étendu" projected CRS can be retrieved at this URL `http://data.ign.fr/id/ignf/crs/NTFLAMB2E`.

3.4 Vocabularies for Geographic Feature Types

Indirect georeferencing of resources on the Web requires reference geographic data on named places and therefore vocabularies for describing feature types and their properties. Therefore, we have chosen to publish a reference dataset on administrative units called GEOFLA®, which is already available in GIS format under an Open Data license. We have also made tests of data conversion and interlinking with another largest dataset on French names places. We have produced and published two vocabularies to describe these datasets, to make sure that all concepts and properties needed would be available. In the GEOFLA® vocabulary[19], 5 classes have been defined: commune, canton, arrondissement, department and region. In the BD TOPO® vocabulary[20] 35 main classes have been defined. They represent the main types of geographic features represented in the BD TOPO® database. In both vocabularies, properties have been defined based on the attributes of their related classes in the databases. The geographic feature types defined as values of attributes "nature" are modeled as instances of `skos:Concept`. SKOS is intensively used to easily group concepts into different schemes (using `skos:hasTopConcept`) and provide semantic relationships (e.g: `skos:broader`, `skos:narrowMatch`) among them. We also provide alignments with Geonames vocabulary, where `topo:Place` is subclass of `gn:S` and `owl:sameAs` linked concepts.[21]

Regarding use cases consuming real-world databases developed using the vocabularies aforementioned, two different applications have been developed. namely *PerfectSchool*[22] and *Equipment*[23]. The former is a mobile application intended to provide useful information on schools in France, while the latter is a facet view by categories of facilities in France, specifically in the city of Toulouse.

[17] `http://data.ign.fr/def/ignf`
[18] A service to lookup CRS in RDF can be found at `http://www.eurecom.fr/~atemezin/ignf-lookup/`
[19] `http://data.ign.fr/def/geofla#`
[20] `http://data.ign.fr/def/topo`
[21] `https://github.com/gatemezing/ign-iswc2014/blob/master/vocabularies/mappingsGeonames.ttl`
[22] `semantics.eurecom.fr/datalift/PerfectSchool/`
[23] `http://semantics.eurecom.fr/datalift/Equipment/`

4 Publishing administrative units (GeoFla)

As a dataset dedicated to administrative units, GEOFLA®is very likely to be reused by other datasets, either by reusing directly its URIs for georeferencing needs, or by reusing its description of administrative units - labels, properties and geometries - for interlinking purposes.

4.1 Data conversion

Geofla is delivered as a set of 4 shapefiles that describe the boundaries and properties of administrative units of mainland France (for CRS reasons, overseas territories are delivered within different shapefiles) : communes, cantons, arrondissements and departements. For the sake of our application, we have generated another shapefile describing regions by aggregating the geometries of the instances of departments based on their region's foreign key value. This dataset is updated every year. Publishing this data in RDF with unique identifiers on the Web will ease the interlinking with some existing datasets describing French boundaries in the wild. We follow a two steps conversion: we use the SHP2RDF module of Datalift to obtain a raw RDF from shapefiles, and the RDF2RDF module of Datalift using a set of SPARQL construct queries[24] for getting a refined RDF datasets using suitable vocabularies.

4.2 URI design policy

One of the requirements to publish data is to have unique ids and stable URIs[25] . Since our legacy databases have unique IDs to refer to the objects, we had to make sure they were unique at Web level. Thus, the base scheme for vocabularies URIs is: `http://data.ign.fr/def/`. Besides, the base schema for identifying a real world resource uses `http://{BASE}/id/`. For example, IGN main buildings are located in the commune with the URI `rgeofla:commune/94067`, corresponding to Saint-Mandé, and `rgeofla:departement/94` corresponds to the department "Val de Marne" to which the commune belongs.

4.3 Interlinking with existing GeoData

We interlinked our datasets with NUTS, DBpedia FR[26] and GADM datasets. SILK [6] is used to interlink the departments in our dataset with departments in DBpedia FR, using labels and INSEE Code. We obtained 93 matches (all correct) while three are missing for the departments 07, 09 and 75[27]. The LIMES tool[28] is then used to perform the rest of the interlinking tasks [7] with the trigrams function based on the labels with restriction to France.

[24] `https://github.com/gatemezing/ign-iswc2014/tree/master/rdf2rdf`
[25] `http://www.w3.org/TR/ld-bp/#HTTP-URIS`
[26] `http://fr.dbpedia.org/`
[27] `https://github.com/gatemezing/ign-iswc2014/tree/master/interlinking/matched`
[28] `https://github.com/AKSW/LIMES.`

– Geofla-RDF with DBpedia FR: **23 252** links obtained. This results show the missing of nearly 13 435 communes not correctly typed in DBpedia FR as `Spatial Feature` or `Place`, or not having a French Wikipedia entry.
– Geofla-RDF with GADM (8 314 443 features): **70** links obtained: 10 communes, 51 departments and 9 regions. The property `gadm:in_country` is used to restrict the interlinking to France. E.g.: The city of Saint-Alban in Quebec is a commune in France.
– Geofla-RDF with NUTS (316 236 triples): Using a *"naive"* script with `trigrams` function on `geofla:Commune/rdfs:label` and `spatial:Feature/ramon:name` reveal two odd results located in Germany and Switzerland. The latter being the *JURA* and the former named *"Celle"*. In order to remove those odd effects, we add another restrictions based on `ramon:code` by filtering the ones located in France (136 features) . The final matchings give a total of **105** correct links: 14 communes, 75 departments and 16 regions.

The above results show good precision of the matching algorithm (score above 0.98) and a rather low recall value with DBPedia-FR (0.627). The few number of matched entities is likely due to the low coverage of French features in the datasets.

5 Publishing French Gazetteer

In this section, we present some first tests of converting BDTOPO® into RDF and interlinking with LinkedGeoData using LIMES. The results confirm the need for geographic publishers to publish georeference data on the Web.

Data conversion, URIs and Interlinking: Shapefiles are converted into RDF using the same two conversion process as for GEOFLA®. The URIs for each resource follow the pattern: `rtopo:CLASS/ID` for the feature, while `rtopo:geom/CLASS/ID` is used to reference the geometry of the resource. The gazetteer dataset in RDF is part of BD TOPO® database consisting of 1,137,543 triples (103,413 features). We chose LinkedGeoData (LGD) [29] to perform the alignments using the main class `lgdo:Amenity`[30] (5,543 001 triples), as they are closed to the features contained in the gazetteer. We perform the interlinking on the geometries using the `hausdorff` metric of LIMES tool. A total of **654** alignments was obtained above the threshold (0.9). This relatively low number of hits can be explained by the coverage of French data in LGD, and the subset of BDTOPO® used for the interlinking. Table 2 provides details of the alignments with subclasses of Amenity.

6 Opportunities and Challenges

The need for interoperable reference geographic data to share and combine georeferenced environmental spatial information is particularly acknowledged by

[29] `http://linkedgeodata.org/sparql`
[30] `http://linkedgeodata.org/ontology/`

LGD Class	#links matched
lgdo:Shop	252
lgdo:TourismThing	30
lgdo:Craft	3
lgdo:AerowayThing	37
lgdo:AerialwayThing	11
lgdo:EmergencyThing	56
lgdo:HistoricThing	257
lgdo:MilitaryThing	8

Table 2: Interlinking results using the Hausdorff metric of LIMES tool between LinkedGeoData and toponyms in the French Gazetteer

the INSPIRE Directive. For geographic data producers, the benefit of publishing their data on the Web according to Linked Data (LD) principles is twofold. On the one hand, their data are interoperable with other published datasets and they can be referenced by external resources and used as spatial reference data, which would not have been straightforward when published according to spatial data infrastructures (SDI) standards. On the other hand, the use of semantic Web technologies can help addressing interoperability issues which are not solved yet by geographic information standards. Moreover, there are different types of license policies to access data at IGN (e.g.: research purpose, commercial use, access on demand, etc.), with some of them not necessary "open" or free to access: (e.g. BD TOPO®). Although there is a clear understanding of the benefits of publishing and interconnecting data on the web, ongoing investigations on how to combine licenses on datasets are under consideration at IGN. Two solutions are under investigation: (i) different license policies attached to datasets and (ii) the use of a security access mechanism on top of the datasets granting access based on a predetermined configuration on named graphs and resources. According to Linked data principles URIs should remain stable, even if administrative units change or disappear. This implies adapting the data vocabulary in order to handle data versioning and real world evolutions. This issue will be addressed in a future work, as we plan to release a spatio-temporal dataset describing the evolution of communes since the French Revolution. Another issue deals with the automation of the whole publication process, from traditional geographic data to fully interconnected RDF data. The last issue deals with the use of multiple geometries for describing a geographic feature: geometries with different levels of detail, different CRS, different representation choices. This has been superficially addressed in our use case with the use of both polygons and points for representing respectively the surface and the centroid of departments, but should be further investigated for both query answering and map design purposes.

7 Conclusions

In this article, we proposed to contribute to the georeferencing of datasets published on the Web of data by providing two vocabularies designed for representing structured geometries defined with coordinates expressed in any CRS, as well

as referencel geodata resources published under `data.ign.fr`, namely CRS's dataset, the French administrative units dataset and part of the French gazetteer dataset. So far, the French units are interconnected with the French statistical datasets, and reused in metadata fields used by the `www.datalocale.fr` portal for defining the geographic extent of each dataset[31].

Acknowledgments

This work has been partially supported by the French National Research Agency (ANR) within the Datalift Project, under grant number ANR-10-CORD-009.

References

1. A. G. Atemezing and R. Troncy. Comparing Vocabularies for Representing Geographical Features and Their Geometry. In *5th International Terra Cognita Workshop*, Boston, USA, 2012.
2. A. de León, L. M. Vilches, B. Villazón-Terrazas, F. Priyatna, and O. Corcho. Geographical linked data: a Spanish use case. In *International Conference on Semantic Systems (I-SEMANTICS'10)*, Graz, Austria, 2010.
3. J. Goodwin, C. Dolbear, and G. Hart. Geographical Linked Data: The Administrative Geography of Great Britain on the Semantic Web. *Transactions in GIS*, 12:19–30, 2008.
4. INSPIRE Thematic WG CRS and Geographical Grid Systems . Guidelines INSPIRE Specification on Coordinate Reference Systems , 2009. `http://inspire.ec.europa.eu/documents/Data_Specifications/INSPIRE_Specification_CRS_v3.0.pdf`.
5. International Organization for Standardization . ISO 19125-1, Geographic information- Simple feature access - Part 1: Common architecture, 2004.
6. A. Jentzsch, R. Isele, and C. Bizer. Silk-generating rdf links while publishing or consuming linked data. In *Poster at the International Semantic Web Conference (ISWC2010), Shanghai*, 2010.
7. A.-C. Ngonga Ngomo. Orchid - reduction-ratio-optimal computation of geo-spatial distances for link discovery. In *Proceedings of ISWC 2013*, 2013.
8. F. Scharffe, G. Atemezing, R. Troncy, F. Gandon, S. Villata, B. Bucher, F. Hamdi, L. Bihanic, G. Képéklian, F. Cotton, J. Euzenat, Z. Fan, P.-Y. Vandenbussche, and B. Vatant. Enabling linked-data publication with the datalift platform. In *26th Conference on Artificial Intelligence (AAAI-12)*, 2012.
9. P. Vandenbussche, B. Vatant, and L. Rozat. Linked open vocabularies: an initiative for the web of data. In *QetR Workshop, Chambery, France.*, 2011.

[31] `http://www.datalocale.fr/drupal7/dataset/ens-cg33`

Translating Maps and Coordinates from the Great War

Rob Warren[1] and David Evans[2]

[1] Big Data Institute
Dalhousie University
Halifax, Canada
rhwarren@dal.ca
[2] Department of Computing and Mathematics
University of Derby
Derby, UK
d.f.evans@derby.ac.uk

Abstract. Many obsolete coordinate systems used in the past have fallen into disuse. However, the contents of historical documents still refer to these obsolete coordinates and thus translation systems are important in locating historical events. We present a specialized Linked Open Data API constructed to translate obsolete British Trench Map coordinates from the Great War into modern WGS84 coordinates. We report on the design of the API, the construction of the triple structures used to answer queries and the methods used to enrich query results while ensuring network performance. Several use cases for the API are presented and we close with a discussion of our experiences in linking to external data providers.

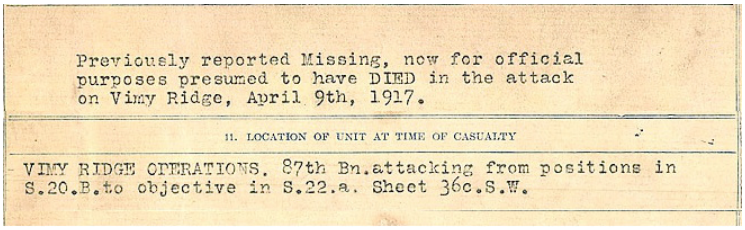
Keywords: Linked Geo Data, Coordinates Translation, Great War

1 Introduction

With the centenary of the Great War renewed interest has been shown in the archival records of this conflict. British and Commonwealth forces used a special coordinate system known as the British Trench Map Coordinate system, which was invented to support military operations on the Western Front.

An example in Figure 1a is an extract from the Circumstances Of Death register of Private John Richard Aaron who went missing during the Battle of Vimy Ridge in 1917. The document records the starting point 36C.S.20.b (the left box in Figure 1b) of the attack and the location of the intended objective at 36C.S.22.a (right box in Figure 1b). His body was never recovered and his remains are likely still there today.

For a number of years translating these coordinates into a modern location could only be done by locating a physical copy of trenchmap sheet 36C, re-projecting the map based on known landmarks and only then could the specific squares S.20.b and S.22.a be geo-located.


Previously reported Missing, now for official
purposes presumed to have DIED in the attack
on Vimy Ridge, April 9th, 1917.

11. LOCATION OF UNIT AT TIME OF CASUALTY

VIMY RIDGE OPERATIONS. 87th Bn.attacking from positions in
S.20.B.to objective in S.22.a. Sheet 36c.S.W.

(a) Circumstances of death of John Aaron

(b) Jumping off position at Squares 36C.S.20.b (left) and objective square 36C.S.22.a (right).

Fig. 1: Linking locations as recorded in archival material to their current locations.

In the context of a Linked Open Data project on the Great War called Muninn[3] it became necessary to be able to translate these coordinates back and forth on a large scale. This paper reports on the design of a Linked Open Data API[4] capable of translating British Trench Map coordinates of the Great War to and from modern WGS84 coordinates.

This paper is organized as follows: we begin with a short introduction to Trench Map Coordinates and the work done to rebuild the mapping system of the time. A brief related work section is then followed by a description of the API functions, the underlying mapping ontology and the data enrichment strategies used by the API. We report on some experimental results obtained in the construction and operation of the API and close with some opportunities that were identified during the course of this work.

2 Problem definition

With the invasion of Belgium by the Germans in 1914, the official Belgian printing plates for the base country maps were evacuated to England where the Ordnance Survey used them as the basis for a new series of small scale maps.

[3] `http://www.muninn-project.org/`

[4] The actual API is at `http://rdf.muninn-project.org/api/TrenchCoordinates`, with a simple web application at `http://rdf.muninn-project.org/TrenchCoordinates.html`.

These were based on a Bonne projection with a Delambre ellipsoid and used the metric system [6,5,8].

These were then merged with information obtained from the French Government about their network of triangulation stations, magnified large scale maps of France, *Plan Directeur* fire control maps and some manual survey works. The set of sheets thus extended beyond the Belgian borders and into France. For reasons that are lost to history, the decision was made to overlay a grid in Imperial (yard) measurements over the metric projection meaning that in some cases duplicate trench coordinates exist and overlap with others.

The specifics of the coordinate systems are reviewed in other documents [2,14,6] but it consists of an alphanumeric string read left to right with increasing accuracy. Most recorded coordinates result in a 50 yard sided square, through a smaller squaring system of a 5 yard sided square was also used[5]. As an example: the location of a trench coordinate such as 27.L.22.d.6.3 would be a 50 yard sided box with centroid 50.8300, 2.7005.

The origin of the original Belgian projection is important because it is used to calculate a conversion between the Bonne projection and the WGS84 datum. It is purported to be the old Brussels observatory, which moved several times and the exact coordinates of the origin remains a point of contention. Positions officially recorded by Mugnier [10] as 50°25'0.0006",4°22'12.6978" and Winterbotham [14] (see also Close [6]) as 51°10'06.895", 4°22'05.89" are both several kilometres off. A recalculation from the original Belgium Triangulation (1867, [4]) by the Belgium Geographical Institute yielded an origin of of 50°24', 4°22'5.89" and after adjustment using several referenced church steeples, 50°23'57.2418", 4°22'10.0518" currently yields results with an average positional error of less than 0.0001 degrees of latitude and longitude.

One of the interesting elements that has caused not a little amount of frustration on the part of the authors is the uneven precision of the maps and the difficulty in obtaining precise location information for referenced land marks within the maps. The angle of observation of the overhead imagery tends to induce errors when trying to locate church steeples precisely and makes make the resolution of the origin difficult. In any event, it is unclear that a local surveyor would have been of help: the French trigonometric points originally used by the ordnance survey referenced some churches that had since been moved before the war and others that were destroyed and rebuilt after the war.

These inaccuracies and problems reflect both the expected "fog of war" as well as some less-than-comprehensible events such as the faulty transcriptions of the locations of French trigonometric stations by cartographers and carelessness in print shops. Peter Chasseaud [5] reviews some of these events in details which

[5] Owing to the particularities of the British Trench Map Coordinates system, a grid reference can be of a number of different rectangular or square shapes. We use the grid square for convenience in the text.

are at times comical and tragic. Some maps have a grid offset by several thousand yards while others have a grid that is inexplicably printed backwards[6].

All of the different uncertainties with the trench coordinates make for a conversion process that can at times return mathematically and geographically sound transformations with historically inconsistent results, *caveat emptor*.

At maximum accuracy, this system's limit was a square with sides of 5 yards through the 50 yard square is much more prevalent. Depending on the sources used to create new map sheets or update the original Belgian ones, the accuracy of the maps would vary dramatically. A complete update of a map by a ordnance survey team would be expected to be accurate within 20 yards [14].

3 Previous Work

Coordinate translation is a common problem that has been tackled comprehensively by Gerald I. Evenden [7] with his Proj4 package (used within our API). Most recently, Troncy et al. [13] published a Java servlet API to translate automatically across different projection in an automated fashion. From a markup and ontological perspective, the (modern) Ordnance Survey has been publishing Linked Open Data using its own coordinate ontology which has inspired our own. In terms of a representation of geometries, Claus et al. [12] used their own RDF constructs, along with the OGC GeoSPARQL [11] and NeoGeo vocabulary[7] to create the Linked Geo Data representation of OpenStreetMap data.

4 Translation API design and implementation

In this section, we review the API and its uses. We begin by describing the operating modes of the API and then review the LOD structures used in its operations and reporting. We report on some analytical results on the opportunistic enrichment of API results in cases where it has not been requested and discuss our experiences in using different enrichment strategies when explicitly requested by the client.

4.1 Query formation

The primary use case is transforming either a trench coordinate or a WGS84 point to the other representation. This transformation is not completely seamless since we move from a continuous coordinate system to a grid based coordinate system. The query is sent to the API through an inline parameter `q` that can contain either a WGS84 location or partial trench map coordinate.

[6] This was not only a Commonwealth experience, the Central powers used a number of competing reference systems, including one grid that was shared between the 4th and 6th German Armies (See [1]), but referenced differently.

[7] `http://geovocab.org/doc/neogeo.html`

In designing this API, we tried to deal with the most common use cases without making decisions that could affect accuracy. Currently, WGS84 is one of the more widespread coordinate system and fits in well with the expectation of an end-user to be able to input the coordinates into their consumer GPS unit. In future work we will integrate the translation API described by Troncy et al. [13] into the API in order for other coordinate systems to be supported.

While queries are submitted through URL parameters, answers are provided in one of RDF, raw text or XML formats with additional LOD formats supported through the parameter `fmt` or content negotiations headers.

Precision and accuracy were important consideration in the handling of the conversion. A trench map is made from several different sources of mapping information: on site surveys, larger scale maps (1:100,000), fire direction maps and the original 1:40,000 Belgium grid plates. The precision one can expect of the map varies wildly depending on the sources used to create it. The re-projection of large scale maps (1:100,000) down to smaller (1:40,000) scale was performed often at the beginning of the war and in these cases one could expect errors of about 200 yards.

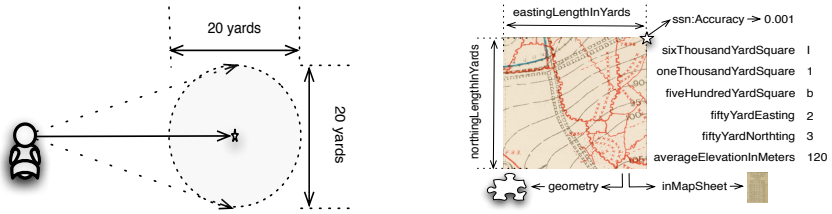
A survey unit making maps from sightings or aerial photographs would achieve a precision of about 20 yards as in Figure 2a. Hence any feature on a map should be expected to be within the area of a circle of a 10 yard radius. Similarly, the estimation of the origin confers an error to the corner points of any trench square. We explain in Section 4.2 in detail the LOD methods used to deal with this, including a separation between the terms that represent a trench coordinate and the corresponding WGS84 feature.

Dealing with the precision of a WGS84 query point remains problematic: A longitude of 5.03 cannot be distinguished from a longitude of $5.0\overline{3}$ or 5.03000000 within most GIS systems. Yet in some cases, the location precision would be a valid means of identifying what size of trench square to return. Currently we transform the point "as-is" and return a square with sides of 5 yards. This is not always ideal and we are experimenting with the use of polygons or precision properties within the query to remedy this.

4.2 Converting coordinates with Linked Open Data

The API is fully enabled to use LOD approaches in reporting conversions to queries and LOD provides a number of tools with which we can avoid precision and accuracy errors from penalizing researchers and users.

Specifically, the conversion is known to introduce errors into the position and there are situations where the grid indicated on an Officer's map was inaccurate. The irony is that this problem is brought upon by the use of technology; officers using these maps during the war would normally not have a problem since map series would be the same across organizational units and registration errors would thus be ignored. Recomputing the actual location that they were referencing depends as much on what feature was drawn on the map where as it does on the mathematical transformation.



(a) Feature accuracy was expected to be within 20 yards. Anecdotally, variations range from 5 yards to more that a 100.

(b) An abridged description of the classes and properties

Fig. 2: A representation of the precision and accuracy problems with coordinates translation

.

The use of RDF/OWL and of the paired ontology[8] was meant to support the following requirements: 1) The location as a British Trench Map Coordinate had to be a separate entity from its location in WGS84. This is both to deal with inaccuracy in the translation and to isolate the location as reported in the documents with the actual location where it occurred. 2) Any ontological statements had to have a minimum of expectation of truthfulness under the previously described problems of precision and accuracy.

The paired ontology contains the different instances of all map sheets used within the coordinates system, the relationships that bind them and the underlying organization of the coordinate system. The ontological structures borrow heavily from the modern British Ordnance Ontologies, Linked Geo Data Ontologies, GeoSPARQL and NeoGeo ontologies. The ontology also provides a convenient repository for the storage of known instances of trench maps so that an imaged version can be quickly located. Representing the geometry is done using a mixture of different style of representation, previously looked at by Auguste et al. [3].

A trench coordinate is a Feature that contains all the information about a trench map location but that is separated from geometry information. The geometry information is added through a property between query and response that always places the geo:Point ogc:sfWithin (or ogc:sfContains) a coordinate square. This ensures the statement is always logically true: because of accuracy issues, we cannot state what the geom:geometry actually is. The benefit of publishing an ontology capable of handling native trench map coordinates is that the locations can be referenced without committing to a specific longitude/latitude translation. This allows authors of semantic web data-sets to use coordinates as a means of locating additional information at that exact location, nearby or within a greater geographic areas. Enforcing the separation of the different co-

[8] `http://rdf.muninn-project.org/ontologies/btmaps`

ordinate system using LOD is what allows people to interchange location data while allowing for uncertainty issues in the translations of the coordinates.

The position of a grid square is communicated using a series of geo:Point and OpenGIS "well known text" instances, one for each of the vertices of the shape. An additional point at the centroid is provided as a convenience for placing labels. The concurrent use of both Point and WKT terms allows for native access from both naive and GeoSPARQL-enabled SPARQL servers.

Calculating the theoretical precision of a transformation from a longitude, latitude point to a grid square is straightforward because the error can be determined from both significant figures and the physical size of the grid square. Documenting the precision is still problematic; there currently exists no standardized way of reporting precision information beyond the terms provided by the Semantic Sensor Ontology[9]. Currently precision information is reported through it and the Provenance Ontology[10].

A method that is used to resolve this issue is the reuse of the reference points used to derive the origin of the Bonne projection. By tracking the accuracy of the computed coordinate transformation against the actual position of the reference points we can get an estimate of the map registration error in the area of a trench coordinate. As with precision information above, reporting this information to the end user is still not standardized from a linked-geo perspective.

Currently, this information is reported using the ssn:Accuracy term from the Semantic Sensor ontology which is still in the incubator stage. In some cases, there is sufficient information about the coordinate systems and maps series that a heat-map of the different probability areas can be reported. This style of data reporting is useful in risk analysis applications, such as located forgotten ammunition depots and an appropriate means of reporting it using linked data is still an open question.

An additional issue in the Trench Map System is that the use of yards for defining squares on a metric map means that the top and bottom of the grid spills into the next map sheet and some coordinates squares overlap. These confusing cases are returned with an additional ogc:overlaps to the alternate square in an attempt to signal corner cases.

4.3 API latency and response enrichment

The API has two use cases: it may be used with applications that are user-facing or it may be used for the batch geo-referencing of an archival collection of maps. These represent extremes of response time requirements. In the former case, the client will want as much enrichment as possible without increasing response time since this saves it from making further requests; in the latter, speed is less of a concern as the process is likely automated. Furthermore, in this case enrichment is likely already contained within the archive catalogue.

[9] http://www.w3.org/2005/Incubator/ssn/wiki/SSN
[10] http://www.w3.org/TR/prov-o/

```
GET /api/TrenchCoordinates?q=57c.i.11.d.5.6 HTTP/1.1
User-Agent: curl/7.22.0 (x86_64-pc-linux-gnu) libcurl/7.22.0
    OpenSSL/1.0.1 zlib/1.2.3.4 libidn/1.23 librtmp/2.3
Host: rdf.muninn-project.org
Accept: application/rdf+xml
```

Fig. 3: Sample API request

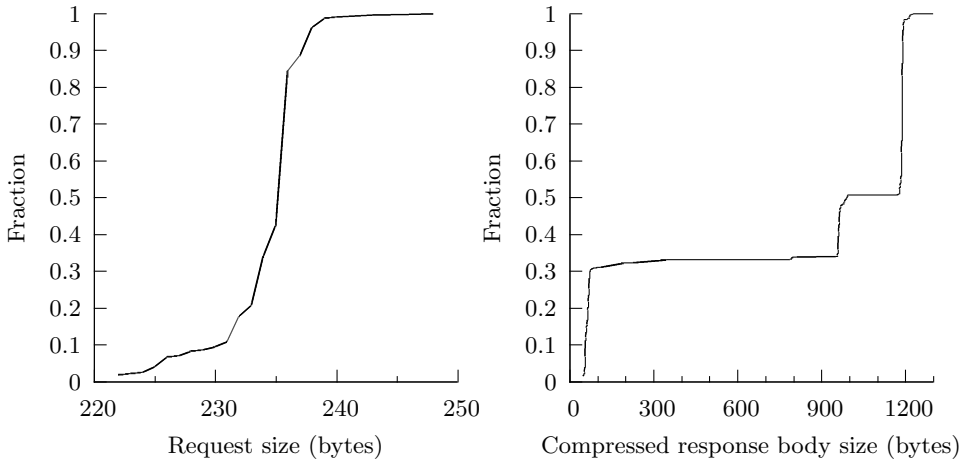
```
HTTP/1.1 200 OK
Date: Thu, 1 Aug 2014 14:10:48 GMT
Server: Apache/2.2.22 (Ubuntu)
Pragma: no-cache
Cache-control: no-cache
Access-Control-Allow-Origin: *
Content-Encoding: gzip
Content-Type: application/x-gzip
Expires: Thu, 1 Aug 2014 14:10:48 GMT
```

Fig. 4: Sample response HTTP headers

Anecdotally, a typical document of this period such as a Regimental War Diary contains about 4 coordinates references per handwritten page. If we assume that the maximum permissible page load time is a generous 5s, requesting a single coordinate should not take longer than 1s. Besides careful provisioning and administration of the server, the network remains the most important factor in meeting this constraint.

If a request and its response each comprise only one network packet, there is no lag from packet reordering or fragmentation and reassembly which is the best scenario. Two questions arise: (i) is an average network packet big enough to hold typical responses, including protocol overhead? and (ii) how much space is left in this packet to enrich the responses with additional practical URIs? In this section we provide answers to these two questions.

A response in one packet As a first step we estimate the expected maximum size of one network packet. Any given path through the Internet is composed of one or more network links. Each of these links has a *maximum transmission unit* (MTU) size which reflects the largest packet that the link can carry. Messages larger than this must be broken into multiple packets. When data traverses several network links en route from sender to receiver, the smallest MTU along the path determines the size of message that may be sent without being fragmented along the way. The *maximum segment size* for a transport protocol like TCP will be this MTU minus protocol overhead. HTTP is used by the API and it runs over TCP, so this MSS value is what interests us. Luckie and Stasiewicz found an MSS of 1460 bytes for about 86% of IPv4 paths (and 1440 for 85% of IPv6 paths) [9].



(a) Cumulative fraction of request sizes

(b) Cumulative fraction of response sizes

Fig. 5: Cumulative fraction of request and response sizes

Figure 3 shows a sample API request. 6111 representative API requests were recorded from the public-facing server over one month and Figure 5a is a plot of cumulative sizes, i.e., the fraction of requests that are at or below a given size (for example, about 40% of requests are at most 235 bytes). All requests were under 300 bytes, so each request that was observed fits into one packet.

Figure 4 shows the headers from a typical API response. These headers amount to 259 bytes (each line must be terminated by the two-byte sequence `0x0d 0x0a`) and with the required blank line leaves 1199 bytes left in an IPv4 packet (1179 for IPv6). The response body is be about 3500 bytes for a basic grid reference but the variant of Lempel-Ziv compression [15] used by the `gzip` tool can reduce this. Figure 5b is a plot of cumulative sizes of 4813 of these compressed responses. (Note that some of the requests used in figure 5a were invalid and so led to no response.) About 70% of these will fit, alongside the protocol overhead of headers, within one packet.

Enriched responses in one packet The primary form of enrichment we have in mind is the inclusion of URIs of "relevant" information, where relevancy is defined by the server based on context. In other words, whenever possible the API will attempt to "round out" a response packet with opportunistic linkages. A number of these are present within the ontology used to track map instances which reference geonames entities and battle locations. In these specific circumstances, the cost of adding extra triples to the response packet has already been amortized while the data may be of benefit to the requesting client.

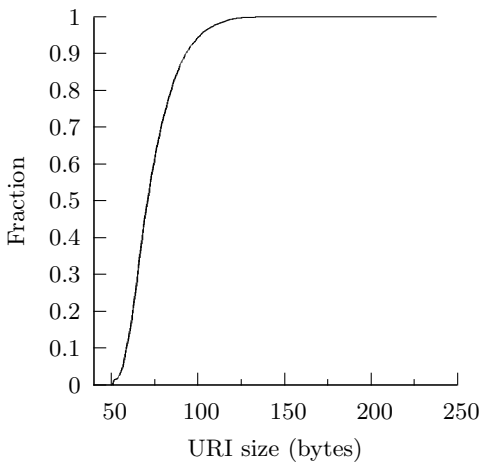


Fig. 6: Cumulative fraction of URI sizes in bytes

To evaluate the extent of enrichment that is feasible whilst retaining the advantages of a single-packet response, we use a set of URIs linking to DBpedia; we believe that this forms a representative set as DBpedia stands at the centre of the Linked Open Data Cloud and is arguably one of the better known data-sets. We obtained a list of 1,566,746 such URIs[11]. Figure 6 shows the cumulative fraction of URIs that are under a certain size; more than 90% of URIs are smaller than 100 bytes. In conjunction with figure 5b, this suggests that about half the time at least one URI can be expected to fit in a single packet alongside the API's response. A nontrivial fraction of requests (about 30%) will leave room for two or three enrichment URIs.

4.4 Explicitly enriching answer terms

In the previous section, we looked at opportunistically enriching the query answer because the cost of doing so was negligible. This is the default behaviour of the API that can be turned off with the enrich=none parameter.

In other cases a request for enrichment of the query results can be made through the enrichment parameter enrich=full. This data can be derived from the trenchmap ontology or external data sources, including geonames for country information or larger data-sets.

The French National Library publishes extensive cataloguing data as part its experimental data portal[12]. We considered querying it as part of our enrichment strategy in that the library contains a number of map holdings of the periods

[11] `http://downloads.dbpedia.org/3.9/en/iri_same_as_uri_en.nt.bz2`, retrieved 2014-07-07

[12] `http://data.bnf.fr/`

before and after the conflict. The problem is that most of the geo-spatial data is still locked within human readable strings, making the geo-location of a map difficult.

Similarly, the German National Library[13] does provide OGC triples for some of its holdings and the API reports them on an opportunistic basis. However, the library lacks a SPARQL server with which to query the data which requires the loading of data into a local SPARQL server for use. Besides data-set size, in this case about the size of the average hard drive in a desktop, there is no means of keeping the local copy synchronized for updates. Given that the average query is likely not to find a document relevant to the locality of interest, the investment in storage and bandwidth is hard to justify.

A promising area of research is in the retrieval of a "working set" of triples that are relevant to the researchers or study's area of interest. Currently, this is exemplified by the non-LOD APIs such as the OSM Overpass[14] that allows clients to retrieve all features within a bounding box.

In the case of the trench map API, it can serve as a retrieval engine by enumerating all known features from the Great War era that are within the area of interest, such as all known features (enrich=full) of Sheet 36C. The advantage of this API feature is that all of the required data is retrieved in one transaction. Usually when a client is faced with a server whose query engine is limited a strategy of flooding the server with multiple queries is used which can results in an overload of the server. If the client interleaves the requests, then the time delay cost due to latency makes the query impractical.

5 Future work and Conclusion

In this paper we have presented an API used for retrieving geo-spatial information reckoned in obsolete military coordinates. The principal job of servers implementing this API is to translate from these coordinates, dating from the Great War, into a modern coordinate system. The issues encountered in processing requests using a Linked Open Data approach where reported on.

There is nothing in the API's design that ties it to the British Trench Map Coordinate System. The Central powers also had their own coordinate systems for the Western Front, including one that was used in two different ways by two different German armies. Given the obvious overlap in operational records, it would be useful to be able to use the API to link the same locations referenced by the different belligerents. We expect such generalizations to be straightforward and valuable to historians.

Finally, the use of LOD provides opportunities to link the geo-spatial information relevant to the API with other information. For example, established social network vocabularies such as Foaf include terms for interests, such as

[13] `http://www.zeitschriftendatenbank.de/de/services/schnittstellen/linked-data/`

[14] `http://wiki.openstreetmap.org/wiki/Overpass_API`

foaf:interest. Given the pointed nature of our geo-spatial data, it may be possible to leverage Linked Open Data to match different historians with interests in specific locations (and contexts) for future collaborations.

Acknowledgements

The authors wish to thank Pierre Voet for his help with the Belgian triangulation and Bill Frost for access to his database of church steeples.

References

1. Imperial War Museum, WW1 Map M-025838.
2. Howard Anderson. *How to find a point on a Great War trench map*. The Western Front Association.
3. Ghislain Atemezing and Raphaël Troncy. Comparing vocabularies for representing geographical features and their geometry. In *11th International Semantic Web Conference, Terra Cognita Workshop*, volume 901, 2012.
4. Institut cartographique militaire. *Triangulation du Royaume de Belgique*. Number v. 2-3 in Triangulation du Royaume de Belgique. Cnophs, 1867.
5. Peter Chasseaud. *Artillery's astrologers : a history of british survey and mapping on the western front 1914-1918*. Naval and Military Press Ltd, March 1999.
6. Charles Close, Lieut.-Colonel Jack Keeling, and et al. Lieut.-Colonel Salmon. British survey on the western front: Discussion. *The Geographical Journal*, 53(4):pp. 271–276, 1919.
7. Gerald I. Evenden. *Cartographic Projection Procedures for the UNIX Environment - A User's Manual*, September 1995.
8. E. M. Jack. *Report on survey on the Western Front 1914-1918*. His Majesty's Stationery Office, London, 1920.
9. Matthew Luckie and Ben Stasiewicz. Measuring path mtu discovery behaviour. In *10th Conference on Internet Measurement*, pages 102–108, 2010.
10. Clifford J. Mugnier. The kingdom of belgium. *Photogrammetric Engineering & Remote Sensing*, pages 956–957, October 1998.
11. OGC. OGC GeoSPARQL - a geographic query language for RDF data. Technical Report OGC 11-052r4, Open Geospatial Consortium, 2012.
12. Claus Stadler, Jens Lehmann, and et al. Linkedgeodata: A core for a web of spatial open data. *Semantic Web Journal*, 3(4):333–354, 2012.
13. Raphaël Troncy, Ghislain Atemezing, and et al. Modeling geometry and reference systems on the web of data. In *W3C Workshop on Linking Geospatial Data*, 2014.
14. H. S. L. Winterbotham. British survey on the western front. *The Geographical Journal*, 53(4):pp. 253–271, 1919.
15. J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Trans. Inf. Theor.*, 23(3):337–343, May 1977.

GeoTriples: a Tool for Publishing Geospatial Data as RDF Graphs Using R2RML Mappings*

Kostis Kyzirakos[1], Ioannis Vlachopoulos[2], Dimitrianos Savva[2], Stefan Manegold[1], and Manolis Koubarakis[2]

[1] Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
{firstname.lastname}@cwi.nl
[2] Department of Informatics and Telecommunications
National and Kapodistrian University of Athens, Greece
{johnvl,dimis,koubarak}@di.uoa.gr

Abstract. A plethora of Earth Observation data that is becoming available at no charge in Europe and the US recently reflects the strong push for more open Earth Observation data. Linked data is a paradigm which studies how one can make data available on the Web, and interconnect it with other data with the aim of making the value of the resulting "Web of data" greater than the sum of its parts. Open Earth Observation data that are currently made available by space agencies such as ESA and NASA are not following the linked data paradigm. Therefore, Earth Observation data and other kinds of geospatial data that are necessary for a user to satisfy her information needs can only be found in different data silos, where each silo may contain only part of the needed data. Publishing the content of these silos as RDF graphs, enables the development of data analytics applications with great environmental and financial value. In this paper we present the tool GeoTriples that allows for the transformation of Earth Observation data and geospatial data into RDF graphs. GeoTriples goes beyond the state of the art by extending the R2RML mapping language to be able to deal with the specificities of geospatial data. GeoTriples is a semi-automated tool that allows the publication of geospatial information into an RDF graph using the state of the art vocabularies like GeoSPARQL and stSPARQL, but at the same time it is not tightly coupled to a specific vocabulary.

Keywords: Linked Geospatial Data, data publishing, GeoSPARQL, stSPARQL

1 Introduction

Lots of Earth Observation (EO) data has become available at no charge in Europe and the US recently and there is a strong push for more open EO data. Linked data is a paradigm which studies how one can make data available on the Web, and interconnect it with other data with the aim of making the value of the resulting "Web of data" greater than the sum of its parts. In the last few years,

*This work has been funded in part by the FP7 project LEO (611141).

linked geospatial data has received attention as researchers and practitioners have started tapping the wealth of geospatial information available on the Web. As a result, the linked open data (LOD) cloud has been rapidly populated with geospatial data. For example, Great Britain's national mapping agency, Ordnance Survey, has been the first national mapping agency that has made various kinds of geospatial data from Great Britain available as open linked data.

Recently, the geospatial semantic web has also started to be populated with EO products (e.g., CORINE Land Cover and Urban Atlas published by project TELEIOS[3]). In general, open EO data that are currently made available by space agencies such as ESA and NASA are not following the linked data paradigm. Therefore, EO data and other kinds of geospatial data that are necessary for a user to satisfy her information needs can only be found in different data silos, where each silo may contain only part of the needed data. Publishing the content of these silos as RDF graphs, enables the development of data analytics applications with great environmental and financial value. With the recent emphasis on open government data in many countries, the development of useful Web applications utilizing EO data and geospatial data in general is just a few SPARQL queries away.

Geospatial data in general and EO data in particular, can come in vector or raster form and are usually accompanied by metadata. Vector data, available in formats such as ESRI shapefiles, KML, and GeoJSON documents, can be accessed either directly or via Web Services such as the OGC Web Feature Service or the query language of a geospatial DBMS. Raster data, available in formats such as GeoTIFF, Network Common Data Form (netCDF), Hierarchical Data Format (HDF), can be accessed either directly or via Web Services such as the OGC Web Coverage Processing Service (WCS) or the query language of an array DBMS, e.g., the array-query language SciQL[4]. Metadata about EO data are encoded in various formats ranging from custom XML schemas to domain specific standards like the OGC GML Application schema for EO products and the OGC Metadata Profile of Observations and Measurements.

Automating the process of publishing linked geospatial data has not been addressed yet. For example, in the wildfire monitoring and management application that we developed in TELEIOS, custom Python scripts were used for publishing all necessary data as linked data. For this reason, we designed and implemented the tool GeoTriples in the context of the EU FP7 project LEO[5]. In this paper we present the tool GeoTriples that allows for the transformation of EO data and geospatial data in various formats, such as data stored in spatially-enabled relational databases and raw files, into RDF graphs. GeoTriples goes beyond the state of the art by extending the R2RML mapping language to be able to deal with the specificities of geospatial data. GeoTriples is a semi-automated tool that allows for the publication of geospatial information into an RDF graph using the state of the art vocabularies like GeoSPARQL [9], but at the same

[3] `http://www.earthobservatory.eu/`

[4] `http://www.sciql.org/`

[5] `http://www.linkedeodata.eu/`

time it is not tightly coupled to a specific vocabulary. The publishing process comprises three steps. First, GeoTriples generates automatically R2RML mappings for publishing data that reside in spatially-enabled databases and raw files (e.g., ESRI shapefiles). Afterwards, the user may edit these mappings according to her needs (e.g., utilize a different vocabulary) and finally GeoTriples processes these mappings for producing an RDF graph.

The organization of the paper is as follows. Section 2 discusses related work. In Section 3 we present the architecture of GeoTriples and our extensions to the R2RML language for the geospatial domain. We discuss how GeoTriples generates automatically R2RML mappings and how they are subsequently proccessed for transforming a geospatial data source into an RDF graph. In Section 4 we present an example where we demonstrate how GeoTriples is currently being used for realizing the precision farming application that is being developed in LEO. Finally, in Section 5 we concludes our work.

2 Related Work

In this section we will present related work on methodologies and tools targeting the transformation of existing data sources into RDF graphs. Much work has been done recently on mapping relational data into RDF graphs. An extensive discussion on mapping relational data into RDF can be found in [8]. In this section we will present in detail two prevailing approaches for mapping relational data into RDF: the direct mapping approach and the R2RML mapping language. Then we discuss some relevant tools that implement these approaches and we finish our discussion with tools that convert geospatial information into RDF graphs.

Direct Mapping of Relational Data to RDF. A straightforward mechanism for mapping relational data to the RDF data model is the *Direct Mapping of Relational Data to RDF*[6] approach that became a W3C recommendation in 2012. According to this approach, relational tables are mapped to classes defined by an RDF vocabulary, while the attributes of each table are mapped to RDF properties that represent the relation between subject and object resources. Identifiers, class names, properties, and instances are generated automatically following the respective labels of the input data. For example, given the table `Address`, the class `<Address>` will be generated, and all tuples will be instances of this class. According to this approach, the generation of RDF data is dictated by the schema of the relational database. This mechanism was initially defined in [2], and [11] is an implementation of such a mechanism.

The Mapping Language R2RML. A language for expressing customized mappings from relational databases to RDF graphs is the *R2RML mapping language*[7] that became W3C recommendation in 2012. R2RML mappings provide the user with the ability to transform existing relational data into the RDF data model, following a structure and a target vocabulary that is chosen by the user.

[6] `http://www.w3.org/TR/2012/REC-rdb-direct-mapping-20120927/`

[7] `http://www.w3.org/TR/r2rml/`

R2RML mappings refer to logical tables to retrieve data from an input database. A *logical table* can be a relational table that is explicitly stored in the database, an SQL view or a valid SQL select query. A triples map is defined for each logical table that will be exported into RDF. A *triples map* is a rule that defines how each tuple of the logical table will be mapped to a set of RDF triples. A triples map consists of a subject map and one or more predicate-object maps. A *subject map* is a rule that defines how to generate the URI that will be the subject of each generated RDF triple. Usually, the primary key of the relation is used for this purpose. A *predicate-object* map consists of predicate maps and object maps. A *predicate map* defines the RDF property to be used to relate the subject and the object of the generated triple. An *object map* defines how to generate the object of the triple, the value of which originates from the value of the attribute of the specified logical table.

R2RML processors are applications that take as input a relational database and R2RML mappings and produce RDF graphs. Some R2RML processors provide a virtual SPARQL endpoint that transparently translate SPARQL queries to SQL queries by taking into account the given R2RML mappings. Alternatively, an R2RML processor can choose to export all relational data as an RDF dump according to the given mappings and then offer SPARQL or a linked data interface over the produced data.

Let us now present some tools that are able to translate raw data sources into RDF following the direct mapping approach, R2RML or a custom mapping language. OpenLink Virtuoso [6], Morph[8] [10], Ultrawrap[9] and D2RQ platform [3, 5] are capable of processing R2RML mappings. They support most of the features of R2RML and can process R2RML mappings both for publishing input data as an RDF graph and for querying the input data by translating a SPARQL query into an SQL query. Triplify [1] is a popular tool that follows a light-weight approach for mapping HTTP-URI requests onto relational database queries, and translating the result into RDF statements.

However, little attention has been paid to the problem of publishing geospatial information in the Semantic Web. Most datasets that contain such information are either generated manually or by semi-automated processes.

LinkedGeoData[10] project focuses on publishing OpenStreetMap[11] data as linked data. Sparqlify[12], developed in the context of this project, is employed for this task, using a proprietary mapping language. It creates mappings of spatial datatypes into RDF but until now, it does not discuss on how to deal with non-relational data.

[8] `https://github.com/jpcik/morph`
[9] `http://capsenta.com/ultrawrap/`
[10] `http://linkedgeodata.org/`
[11] `http://www.openstreetmap.org/`
[12] `http://sparqlify.org/`

The tool Geometry2RDF[13] was the first tool that allows for the conversion of geospatial information that resides in a spatially-enabled relational database into an RDF graph. It takes as input data stored in a spatially enabled DBMS like Oracle Spatial or MySQL, and utilizes the libraries Jena and GeoTools to produce an RDF graph. Geometry2RDF follows the direct mapping approach, and allows the user to configure the properties that connect a URI to the serialization of a geometry and allows for the conversion of the coordinates to the desired coordinate reference system. Geometry2RDF follows the direct mapping approach that is not expressive enough to deal with the specificities of the geospatial domain. For example, since this work has preceded the proposal of GeoSPARQL and stSPARQL, it does not produce RDF graphs according to these vocabularies. For this purpose, the tool TripleGeo[14] was recently developed. TripleGeo is based on Geometry2RDF and allows the generation of RDF graphs that follow the GeoSPARQL vocabulary. However, this tool has the same shortcomings with Geometry2RDF since the mapping process of the input data into an RDF graph that follows the GeoSPARQL vocabulary is hard-coded in the implementation. Thus, significant effort is required for modifying such tools in order to support other vocabularies.

A different approach was followed by [4] where the authors present how R2RML can be combined with a spatially enabled relational database in order to transform geospatial information into RDF. For the manipulation of the geometric information prior to its transformation into RDF, the authors create several logical tables that are based on ad-hoc SQL queries that perform the appropriate pre-processing (e.g., requesting the serialization of a geometry according to the WKT standard). This approach demonstrates the power of utilizing a general-purpose mapping language like R2RML, which is in contrast to other approaches discussed earlier in this section. However, in this work, no automated method for publishing geospatial datasets into RDF is discussed, and dealing with different types of data sources was out of scope.

3 The Tool GeoTriples

In this section we will present in detail the tool GeoTriples that we developed for transforming geospatial data sources into RDF. GeoTriples[15] is an open-source tool that is distributed freely according to the Mozilla Public License v2.0. In this section we will present the architecture of GeoTriples and we will discuss our implementation choices. We will describe in detail how GeoTriples generates automatically R2RML mappings for publishing data that reside in spatially-enabled databases and raw files (e.g., ESRI shapefiles), and how it can process such mappings for producing an RDF graph that follows the GeoSPARQL or any other vocabulary.

[13] `http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/technologies/151-geometry2rdf`

[14] `https://github.com/GeoKnow/TripleGeo`

[15] `http://sourceforge.net/projects/geotriples/`

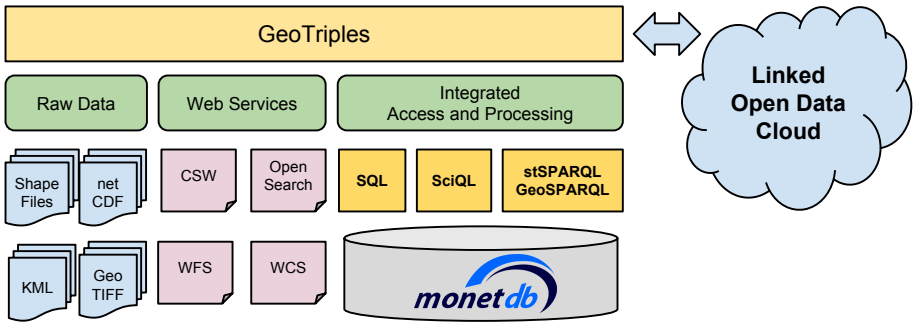


Fig. 1: The abstract architecture of the system GeoTriples

3.1 System Architecture

The abstract architecture of GeoTriples, from a data perspective is shown in Figure 1. GeoTriples takes as input data that are stored in a spatially-enabled database, data that reside in raw files (e.g., ESRI shapefiles), or the results that derive from processing the aforementioned data (e.g., the result of a SciQL query over raster or array data). Regarding the system architecture at a lower level, GeoTriples uses a connector for each type of input data in order to transparently access and process the input data. As Figure 2 depicts, GeoTriples comprises two main components: the mapping generator and the R2RML processor. The mapping generator takes as input a data source and creates automatically an R2RML mapping that transforms it into an RDF graph. The generated mapping is enriched with subject and predicate-object maps, in order to take into account the specifities of geospatial data and cater for all transformations that are needed to produce an RDF graph that is compliant with the GeoSPARQL vocabulary. To accomplish this task, we extend R2RML mappings to allow the representation of a transformation function over input data. Afterwards, the user may edit the generated R2RML mapping document to comply with her requirements (e.g., use a different vocabulary).

Implementation Choices. One of the main choices we had to make during the design of GeoTriples was to choose which RDB2RDF framework to extend. We chose to extend the D2RQ platform which seemed to be the most mature system for publishing relational data into RDF. It provides a mechanism for generating and processing R2RML mappings for a variety of relational databases that are accessible via JDBC, like MonetDB, PostgreSQL and Oracle. D2RQ provides preliminary support for translating SPARQL queries into SQL queries given some R2RML mappings. However, given the recent results of the system Morph [10], we will consider utilizing Morph as the R2RML processor of GeoTriples in the future.

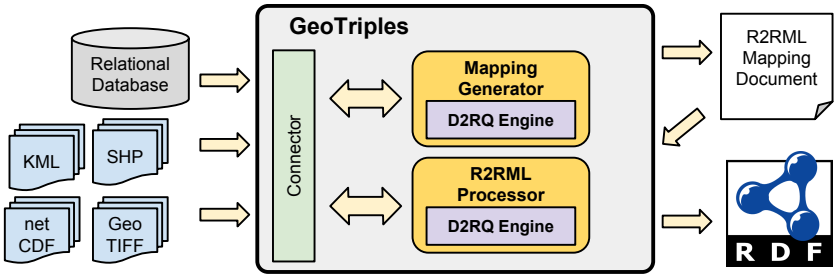


Fig. 2: The architecture of the system GeoTriples

3.2 Transforming Geospatial Data into RDF graphs using GeoTriples

In this section we will present the main functionality of GeoTriples. First we will show the extended R2RML mappings that are produced automatically by GeoTriples that take into account the spatial dimension of the input data, and then we will show how these mappings are processed subsequently by GeoTriples in order to generate an RDF graph.

Automatic Generation of R2RML Mappings. Much work has been done recently on extending RDF to represent and query geospatial information. The most mature results of this work are the data model stRDF and the query language stSPARQL [7] and the OGC standard GeoSPARQL. GeoSPARQL is an OGC standard for the representation and querying of geospatial linked data. GeoSPARQL defines much of what is required for such a query language by providing vocabulary (classes, properties, and functions) that can be used in RDF graphs and SPARQL queries to represent and query geospatial data. The top level classes defined in GeoSPARQL are `geo:SpatialObject` that has as instances everything that can have a spatial representation and `geo:Feature` that represents all features and is the superclass of all classes of features that the users might want to define. To represent geometric objects, the class `geo:Geometry` is introduced by GeoSPARQL. Additional vocabulary is also defined by GeoSPARQL for asserting and querying information about geometries.

Given a spatially-enabled database or a raw file that contains geometric information, GeoTriples generates an R2RML mapping document. Let us take for example the Natura 2000 dataset of Germany that contains information about protected areas in Germany and is distributed in the form of an ESRI shapefile. More details on this dataset can be found in Section 4. Conceptually, each geometric object stored in the ESRI shapefile should be an instance of the class `geo:Geometry`, and all non-geometric attributes that characterize each geometry are thematic attributes of the corresponding feature. Following this modeling approach, we generate an instance of the class `geo:Feature` and an instance of

the class `geo:Geometry` for each geometric object stored in the ESRI shapefile. Geometric and non-geometric attributes that appear at the ESRI shapefile are assigned accordingly to the appropriate instances. Part of the R2RML mapping that is generated automatically by GeoTriples to represent the features stored in the Natura 2000 ESRI shapefile is the following:

```
_:natura
    rr:logicalTable [ rr:tableName "‘natura‘"; ];
    rr:subjectMap [
            rr:class geo:Feature;
            rr:template "http://data.example.com/natura/Feature/id/{‘gid‘}"; ];

    rr:predicateObjectMap [
            rr:predicate nato:has_SITECODE;
            rr:objectMap [ rr:datatype xsd:string;
                       rr:column "‘SITECODE‘"; ];
    ];
    rr:predicateObjectMap [
            rr:predicate geo:hasGeometry ;
            rr:objectMap [
                rr:parentTriplesMap _:natura_geometry;
                rr:joinCondition [
                        rr:child  "gid";
                        rr:parent "gid"; ]; ]; ].
```

This mapping document contains a triples map that describes how instances of the class `geo:Feature` are constructed. All thematic information that is stored in the ESRI shapefile is assigned to the generated feature and a link between the feature and its geometry is also generated. However, the notion of a primary key is not defined for ESRI shapefiles. Each ESRI shapefile though is accompanied by a relational table in dBASE format that stores infromation about the geometries, so we define as a unique identifier for each geometric object the respective row identifier in the dBASE table. In the example above we represent this information as an extra attribute with name 'gid'.

Part of the R2RML mapping that is generated automatically by GeoTriples to represent the geometries stored in the Natura 2000 ESRI shapefile is the following:

```
_:naturaGeometry
    rr:logicalTable [ rr:tableName "‘natura‘"; ];
    rr:subjectMap [
        rr:class geo:Geometry;
        rr:template "http://data.example.com/natura/Geometry/id/{‘gid‘}"; ];

    rr:predicateObjectMap [
        rr:predicate geo:dimension;
        rr:objectMap [
```

```
rrx:transformation [
    rrx:function geof:dimension;
     rrx:argumentMap (
        [rr:column "‘Geom‘"] ); ]  ]; ].
```

This mapping document contains a triples map that describes how instances of the class `geo:Geometry` are constructed. All geometric information that is stored in the ESRI shapefile is assigned to the generated geometry. In addition, object maps define that geospatial functions like `geof:dimension` are applied to the serialization of each geometric object in order to produce the values that will appear at the object part of the corresponding triple.

Notice that in the above example we extended the definition of an object map by allowing it to be the RDF term obtained by applying a transformation on the source data. Each transformation defines the SPARQL built-in function or the SPARQL extension function to be invoked, using as an argument the sequence of RDF terms that are produced by the respective term maps.

Processing of R2RML mappings for producing RDF graphs. R2RML mappings usually consist of two triples maps; one for handling thematic information and one for geospatial information. The triples map that handles thematic information defines a logical table that contains the thematic attributes and a unique identifier for the generated instances. The latter could be either the primary key of the table in case the input data is a relational database or a row number in the dBASE table of an ESRI shapefile. Combined with a URI template, the unique identifier is used to produce the URI that serve as subjects of the produced triples. The predicate object maps are also processed accordingly in order to define RDF properties the value of which originate from the value of the column of the thematic logical table.

The triples map that handles the geospatial information of the input data source, defines a logical table with unique identifier similar to the thematic one. However, according to the type of the data source, the definition of this logical table may vary. For instance, in the case of a relational database, it is defined by providing an appropriate SQL query that uses spatial functions provided by spatially enabled relational backend (e.g. utilize the function `ST_Dimension`). If the input source is an ESRI shapefile, then GeoTriples will perform such transformations on the fly by evaluating the SPARQL extension function using the JTS Topology Suite.

4 Linked geospatial data in the precision farming application of LEO

Let us now give an example that demonstrates how GeoTriples is being used in LEO for the development of a precision farming application. The aim is to develop a precision farming application that combines traditional geospatial data

with linked geospatial data for enhancing the quality of precision farming activities. Precision farming is a concept based on observing the heterogeneity within an agricultural field and providing information about cultivating each part of the field according to the characteristics of the soil. Precision farming aims to solve numerous problems for farmers such as the minimization of the environmental pollution by fertilizers. For dealing with this issue, the farmers have to comply with many legal and technical guidelines that require the combination of information that resides in diverse information sources. In this section we present how linked geospatial data can form the knowledge base for providing solutions for this problem. We published the following datasets as RDF graphs using GeoTriples in order to use them in the precision farming application.

Talking Fields[16] is a precision farming project aiming to increase the efficiency of agricultural production via precision farming by means of geo-information services integrating space and ground-based assets. Currently, TalkingFields produces products for improved soil probing using satellite-based zone maps, and provide services for monitoring crop development through provision of biomass maps and yield estimates.

Natura 2000[17] is an ecological network designated under the Birds Directive and the Habitats Directive which form the cornerstone of Europe's nature conservation policy. National authorities submit a standard data form that describes each site and its ecology in order to be characterized as a Natura site.

OpenStreetMaps (OSM)[18] is a collaborative project for publishing free maps of the world. OSM maintains a community-driven global editable map that gathers map data in a crowdsourcing fashion. LinkedGeoData (LGD)[19], is a project focused on publishing OSM data as linked data.

LGD data is the only dataset that is already published as linked geospatial data. For the rest datasets, we design an appropriate ontology that follows the GeoSPARQL vocabulary. Afterwards, we use GeoTriples in order to produce the R2RML mappings that dictate the process of generating the desired RDF output. Finally, we use the R2RML processor of GeoTriples for translating the input data into RDF graphs. The produced data are then stored into an appropriate geospatial RDF store. We chose to utilize our own geospatial RDF store Strabon[20] [7].

Let us now see an example query that can provide a precision farming application with information about the parts of agricultural fields that are either close to a river or are located within a protected area. This information allows the precision farming application to take into account legal restrictions regarding distance requirements when preparing the prescription maps that the farmers will utilize afterwards.

[16] `http://www.talkingfields.de/`

[17] `http://ec.europa.eu/environment/nature/natura2000/access_data/index_en.htm`

[18] `http://www.openstreetmap.org`

[19] `http://linkedgeodata.org`

[20] `http://www.strabon.di.uoa.gr/`

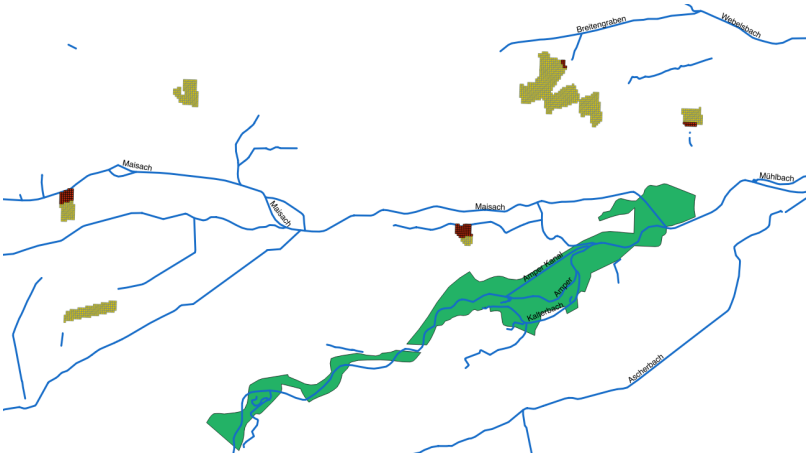


Fig. 3: Visualization of query results

Example. Select all parts of agricultural fields that are close to a river or are located inside a protected area.

```
SELECT  distinct ?field_name ?river_name ?cell_wkt
WHERE {{?river     rdf:type osmo:River ;
                   osmo:hasName ?river_name ;
                   geo:hasGeometry ?river_geo .
        ?river_geo geo:asWKT ?river_wkt .
        ?field     rdf:type tf:Field ;
                   tfo:hasFieldName ?field_name ;
                   tf:hasRasterCell ?cell .
        ?cell      geo:hasGeometry ?cell_geo ;
        ?cell_geo  geo:asWKT ?cell_wkt .
        FILTER(geof:distance(?river_wkt ,
                             ?cell_wkt, uom:meter) < 100) .
       } UNION {
        ?field     rdf:type tf:Field ;
                   tfo:hasFieldName ?field_name ;
                   tf:hasRasterCell ?cell .
        ?cell      geo:hasGeometry ?cell_geo ;
        ?cell_geo  geo:asWKT ?cell_wkt .
        ?nat       rdf:type nat:NaturaArea ;
                   geo:hasGeometry ?nat_geo .
        ?nat_geo   geo:asWKT ?nat_wkt .
        FILTER(geof:contains(?nat_wkt, ?cell_wkt)}}
```

This above query uses the `geo:distance` function whose arguments are the serializations of two geometries and a URI that identifies a unit of measurement. The result of this metric function is the minimum distance between these two

geometric objects. This filter identifies the parts of a field that are close to a river, so the precision farming application should adjust the fertilizer/pesticide usage accordingly. The second part of the query identifies the parts of a field that resides inside a protected, thus the precision farming application should adjust the fertilizer/pesticide usage accordingly. Figure 3 depicts protected areas, rivers, agricultural fields and the parts of the agricultural fields that are close to a river.

5 Conclusions

In this paper we presented the tool GeoTriples that transforms geospatial data that reside in a spatially-enabled DBMS or raw files into RDF graphs, by extending the R2RML mapping language. This allows GeoTriples to transform geospatial data into RDF graphs without being tightly coupled to a specific vocabulary. We plan to extend GeoTriples to support numerous types of vector data formats such as KML, as well as raster formats like GeoTIFF and netCDF.

References

1. S. Auer, S. Dietzold, J. Lehmann, S. Hellmann, and D. Aumueller. Triplify: Lightweight Linked Data Publication from Relational Databases. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 621–630, New York, NY, USA, 2009. ACM.
2. T. Berners-Lee. Relational Databases on the Semantic Web. http://www.w3.org/DesignIssues/RDB-RDF.html.
3. C. Bizer and A. Seaborne. D2RQ-treating non-RDF databases as virtual RDF graphs. In *Proceedings of the 3rd international semantic web conference (ISWC2004)*, volume 2004, 2004.
4. K. Chentout and A. A. Vaisman. Adding Spatial Support to R2RML Mappings. In *OTM Workshops*, volume 8186 of *Lecture Notes in Computer Science*. Springer, 2013.
5. R. Cyganiak, C. Bizer, J. Garbers, O. Maresch, and C. Becker. The D2RQ Platform. http://d2rq.org/.
6. O. Erling and I. Mikhailov. RDF Support in the Virtuoso DBMS. In S. Auer, C. Bizer, C. Müller, and A. V. Zhdanova, editors, *CSSW*, volume 113 of *LNI*, pages 59–68. GI, 2007.
7. K. Kyzirakos, M. Karpathiotakis, and M. Koubarakis. Strabon: A semantic geospatial dbms. In P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. X. Parreira, J. Hendler, G. Schreiber, A. Bernstein, and E. Blomqvist, editors, *International Semantic Web Conference (1)*, volume 7649 of *Lecture Notes in Computer Science*, pages 295–311. Springer, 2012.
8. K. Kyzirakos, S. Manegold, I. Vlachopoulos, D. Savva, M. Koubarakis, C. Nikolaou, S. Giannakopoulou, P. Smeros, and B. Valentin. Data models and languages for mapping EO data to RDF. Deliverable D2.1, LEO - FP7 ICT Project.
9. OGC. GeoSPARQL - A geographic query language for RDF data, November 2012.
10. F. Priyatna, Ó. Corcho, and J. Sequeda. Formalisation and experiences of R2RML-based SPARQL to SQL query translation using Morph. In C.-W. Chung, A. Z. Broder, K. Shim, and T. Suel, editors, *WWW*, pages 479–490. ACM, 2014.
11. D. Steer. SquirrelRDF, 2006. `http://jena.sourceforge.net/SquirrelRDF/`.

Semantic Sensor Networks 2014

7th International Workshop on Semantic Sensor Networks
Riva del Garda - Trentino, Italy, 20 October 2014

Introduction

The number of IP connected devices will be nearly three times the global population by 2016 and a growing amount of Internet traffic is originating with non-PC devices. Many of these devices can act as sensors and there may be trillions of fixed sensors by 2020. Such sensors are geographically distributed and are capable of forming ad hoc networks, with nodes expected to be dynamically inserted and removed. This diverse, changing environment provides many interesting challenges.

While frameworks such as the Sensor Web Enablement (SWE) standards, developed by the Open Geospatial Consortium, provide some interoperability, semantics is increasingly seen as a key enabler for integration of sensor data and broader Web information systems. Analytical and reasoning capabilities afforded by Semantic Web standards and technologies are considered important for developing advanced applications that go from capturing observations to recognition of events and ultimately developing comprehensive situational awareness. Defence, transportation, global enterprise, and natural resource management industries are leading the rapid emergence of applications in commercial, civic, and scientific operations that involve sensors, web services and semantics. Semantic technologies are often proposed as important components of complex, cross-jurisdictional, heterogeneous, dynamic information systems. The needs and opportunities arising from the rapidly growing capabilities of networked sensing devices are a challenging case.

SSN 2014 aims to provide an inter-disciplinary forum to explore and promote the technologies related to a combination of the semantic web, sensor networking and sensors in the Internet of Things. Specifically, to develop an understanding of the ways semantic web technologies can contribute to the growth, application and deployment of large-scale sensor networks on the one hand, and the ways that sensor networks can contribute to the emerging semantic web, on the other.

Organization

Semantic Sensor Networks 2014 was organized in conjunction with the International Semantic Web Conference 2014 at Riva del Garda, Italy.

Program Committee Chairs

Michael Compton (CSIRO Computational Informatics, Australia)
Krzysztof Janowicz (University of California, Santa Barbara, USA)
Kerry Taylor (CSIRO Computational Informatics, Australia)

Advisors

Amit Sheth (Kno.e.sis, Wright State University, Dayton, OH, USA) Manfred Hauswirth (Digital Enterprise Research Institute, National University of Ireland, Galway, Ireland)

Program Committee

Prof Thomas Meyer (Meraka Institute, CSIR, South Africa)
Dr Payam Barnaghi (University of Surrey, UK)
Mr Pramod Anantharam (Wright State University, USA)
Dr Jean-Paul Calbimonte (EPFL, Switzerland)
Dr Danh Le Phouc (Insight, National University of Ireland, Galway, Ireland)
Dr Deshendran Moodley (University of KwaZulu-Natal, South Africa)
Dr Kevin Page (Oxford e-Research Centre, University of Oxford, UK)
Prof Franz Baader (TU Dresden, Germany)
Dr Oscar Corcho (Universidad Politécnica de Madrid, Spain)
Dr Alasdair Gray (Heriot-Watt University, UK)
Dr Josiane Xavier Parreira (National University of Ireland, Ireland)
Dr Cory Henson (Bosch Research and Technology Center, Pittsburgh, USA)

XGSN: An Open-source Semantic Sensing Middleware for the Web of Things

Jean-Paul Calbimonte, Sofiane Sarni, Julien Eberle and Karl Aberer

Faculty of Computer Science and Communication Systems, EPFL, Switzerland.
firstname.lastname@epfl.ch

Abstract. We present XGSN, an open-source system that relies on semantic representations of sensor metadata and observations, to guide the process of annotating and publishing sensor data on the Web. XGSN is able to handle the data acquisition process of a wide number of devices and protocols, and is designed as a highly extensible platform, leveraging on the existing capabilities of the Global Sensor Networks (GSN) middleware. Going beyond traditional sensor management systems, XGSN is capable of enriching *virtual sensor* descriptions with semantically annotated content using standard vocabularies. In the proposed approach, sensor data and observations are annotated using an ontology network based on the SSN ontology, providing a standardized queryable representation that makes it easier to share, discover, integrate and interpret the data. XGSN manages the annotation process for the incoming sensor observations, producing RDF streams that are sent to the cloud-enabled Linked Sensor Middleware, which can internally store the data or perform continuous query processing. The distributed nature of XGSN allows deploying different remote instances that can interchange observation data, so that virtual sensors can be aggregated and consume data from other remote virtual sensors. In this paper we show how this approach has been implemented in XGSN, and incorporated to the wider OpenIoT platform, providing a highly flexible and scalable system for managing the life-cycle of sensor data, from acquisition to publishing, in the context of the semantic Web of Things.

1 Introduction

From wearable devices for health monitoring to geospatial and environmental sensors, we are surrounded by objects or things which are susceptible to be present in the Web, in one way or another. Sensed data on the web is a need and a reality in many real-life use cases and scenarios nowadays. The gap between the real and virtual world is narrowing and there is an increasing necessity to identify everyday life entities in the Web, and let them interact among them, as well as with real people. Many of these challenges have converged towards concepts such as the Internet of Things and the Web of Things, which have gathered enormous attention from academia and the industry [3].

However, when comes the time to expose these data in the Web, there are several problems that data providers may encounter on the way. One is the heterogeneity of the data sources. Starting from the devices themselves, there is

an enormous range of gadgets and equipment with different capabilities, accuracy, range or frequency. There also exist numerous possible IoT protocols and technologies that devices can use to publish data to the Web (CoAP, XMPP, MQTT, DDS, etc.) each targeting different use cases. Many of these challenges have been addressed in previous years from different perspectives [4]. Through a middleware system, applications and users may access data from interconnected objects and things, hiding the internal communication and low-level acquisition aspects. As an example, the GSN middleware has already provided an extensible protocol-agnostic mechanism to acquire data from sensing devices, using configurable wrappers [1], and implementing some of these protocols.

However, these technical difficulties at the lower layers are only the tip of the iceberg, considering that even if they are addressed by platforms such as GSN, there is still a large heterogeneity problem when the data that is sensed needs to be interpreted and understood. For example, the number of possible *observed properties* that may be sensed by an entity, such as humidity, radiation, soil moisture, location detection, etc. can include almost any type of phenomenon or event in the surrounding world. Even if these elements can be abstracted in a domain-model, there are also different ways of exposing and publishing the data in the web, through different formats, under different data models and using different service abstractions. In the end, in many cases the result is a use-case-tailored system that gathers data from a particular set of sources, and exposes them using some ad-hoc data model, creating yet-another isolated silo of data in the web, with very few possibilities of re-use or integration.

One of the ways to tackle these heterogeneity issues is by following a semantics-based approach. Using semantically rich models (ontologies which can be extended for a particular use case), a number of systems [21, 19] have shown how very uneven data sources can be shared and be mutually understandable, while following emerging standards and principles such as Linked Data [7]. In the more specific case of sensor data, specific ontologies and vocabularies such as the SSN (Semantic Sensor Network) Ontology [10] have been created by the community, and have been adopted in a number of projects already [6, 14, 9, 16, 20]. Existing standards for publishing and accessing semantically annotated data (SPARQL[1], Linked Data Platform[2], etc.) are gaining adoption and establishing best practices for sharing data.

In this paper we describe XGSN, a middleware solution that handles the lifecycle of *virtual sensors* (devices, objects or people observing properties around them), providing semantic annotations for them and the observation data that they produce. The key idea is to provide an end-to-end semantic-enabled platform for IoT data management, in which XGSN plays the role of a fully distributed data acquisition middleware with semantic annotation capabilities. We describe the architecture of this system and its implementation, emphasizing on the distributed data processing that allows XGSN to produce different layers of aggregated observations. XGSN extends the successful Global Sensor Net-

[1] W3C Recommendation SPARQL 1.1 http://www.w3.org/TR/sparql11-query/
[2] W3C Candidate Recommendation LDP: http://www.w3.org/TR/ldp/

works [1] system with the semantics-aware capabilities described in this paper, and is available as an open-source package that can be used and extended. The existing community of developers and users inherited from GSN positions this software project as one of the most comprehensive and extensible tools for IoT data management, as it has been shown in several real-life deployments and environmental scientific research. XGSN is a ready-to-use system[3], also available as part of the OpenIoT platform[4], and has also been integrated with the Linked Sensor Middleware (LSM) [16], showing that it can be plugged to an RDF-enabled data store. The remainder of the paper is structured as follows: In Section 2 we describe the general approach of XGSN. Then we present the ontology management aspects and annotation process in Section 3. The architecture is described in Section 4 and the distributed virtual sensor management and experimentation in Section 5. We discuss the related work in Section 6 before concluding.

2 The XGSN Approach for Semantic Data Management

XGSN is built as an extended fork of the GSN middleware [1], which already implemented pluggable sensor data acquisition mechanisms, combined with a distributed stream processing layer. GSN is an inherently decentralized system where different instances can exist in a distributed deployment (see Figure 1), and interchange observation data as needed. The distribution can be based on geographical, economic, privacy or scalability constraints, and each instance can expose a number of different *virtual sensors*. These virtual sensors can be logical abstractions of one or more real sensors or objects or any entity that captures data. They can also be aggregators or filters applied to other virtual sensors, which can be deployed locally or remotely. The interface between devices, sensors or inter-connected *things* and a virtual sensor is a *wrapper*, of which different implementations can co-exist. Different wrappers are already available in the system (e.g. UDP, serial, HTTP, etc.) and creating a new one is generally a simple extension task. Once the data is captured by the wrapper, GSN also provides an extendable processing layer which can be programmed to store the observation data, annotate it, apply correction algorithms over it, etc.

Although GSN already dealt with the problem of handling heterogeneity at the device and acquisition level, it was not able to provide higher level abstractions over the virtual sensors, so that applications could interpret and reuse the data without an external entity deciphering it. In XGSN we follow a semantics-based approach, annotating the virtual sensors with relevant metadata using an extension of the SSN ontology. Two main types of semantic annotations have been added in XGSN. The first are metadata annotations, related to sensors, sensing devices and their capabilities[5], which could not be described before in GSN. These are typically linked to the virtual sensors declared in an XGSN

[3] GSN: `http://gsn.epfl.ch`
[4] OpenIoT: `http:/openiot.eu`
[5] Related to the Measuring and Measuring Capability modules in the SSN ontology

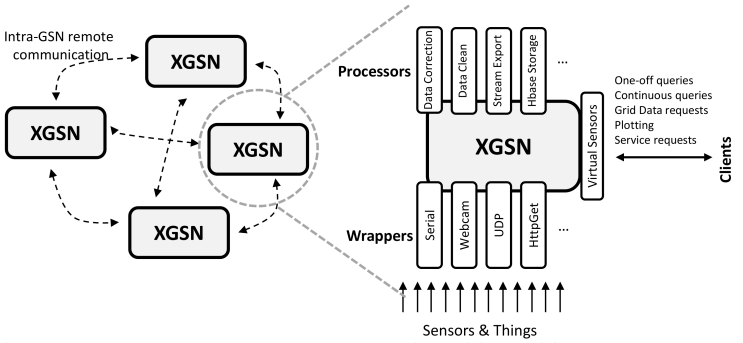


Fig. 1: GSN high level architecture, also applicable for XGSN. XGSN instances may interchange observation data remotely. Each one acquires data through a set of wrappers, and offer continuous data handling capabilities through extensible processors.

instance: e.g. describe the sensing device that produces the data in a particular virtual sensor, its location, the type of observation it produces, the responsible person or organization, the source type, etc. The other type of annotations are related to the observations or measurements produced continuously by the sensors. This includes the semantic information that describes the time and context when the observation happened, the observed property, unit, the values themselves, etc. We will see in Section 3 how these metadata and observation annotations can be exposed as Linked Data using an RDF-enabled cloud system such as LSM.

These explicit semantics in the virtual sensor representation facilitate the tasks of discovery and search in an IoT environment. Also for actual observations, XGSN can provide different levels of semantic annotations to them, depending on the type of virtual sensor that is exposed. For example, in an air quality scenario, XGSN can annotate each measurement made by a sensor (including value, unit, data type, etc.) as an observation of a property (e.g. NO_2), as depicted in Figure 2. However, for some use cases low level annotations are not useful or relevant, so XGSN can aggregate, filter or process several observations over time or space. This will produce indicators in a higher level virtual sensor, each of which can be annotated with even higher-level concepts of a domain ontology (e.g. a "low air quality" observation). Furthermore, even more complex correlations, and processing including external data sources or data from other XGSN instances, can lead to annotations that denote actionable and human-comprehensible concepts like alerts or activities.

In order to make this possible, XGSN relies on ontologies for sensor and observation representation, with three main extensibility points: at the model, data acquisition and processing levels, as we will detail next.

3 Ontologies and Annotation in XGSN

The basis of the abstract model used by XGSN for sensing entities and observations in the web of things, is the SSN ontology[6]. This ontology is not limited

[6] SSN Ontology: `http://purl.oclc.org/NET/ssnx/ssn`

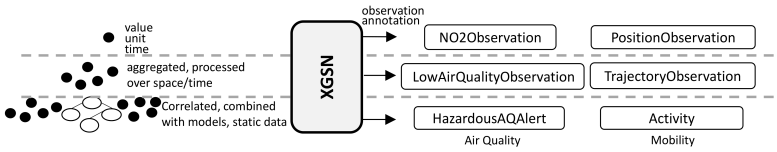


Fig. 2: XGSN annotations at different abstraction levels. From annotation of particular observations to high level concepts that aggregate, summarize or combine more data sources, the processing capabilities of XGSN allow defining annotation at different levels depending on the IoT use cases (i.e. air quality, mobility use cases, etc.)

to sensors thought as devices like thermistors or wind anemometers but more generally to any entity capable of observing a property of a feature of interest [10]. Therefore, interconnected objects and things can provide information about the events, facts and observations surrounding them. The SSN ontology is designed to be extended depending and according to the domain of use. With this in mind, XGSN takes this model as a the core of the metadata and observation annotations. We can see a summary of the main concepts of the ontology in Figure 3 along with extensions and examples of domain-specific vocabularies. The first important extension point is at the sensor level. Any virtual sensor (independently of being a device or other type of entity) is annotated in XGSN as a instance of ssn:Sensor[7] or a sub-class of it, like a thermistor or a capacitive bead. Individuals can also be sensors, as they can also observe events and observations surrounding them. The other two key extension points are related to the *observed property* and to which *feature of interest* it is associated. XGSN requires each sensor to observe at least one property, which can be a domain specific instance, such as the cf-prop:air_temperature of the dim:Temperature quantity in Figure 3. Each of these observed properties of a sensor corresponds to a field defined in a XGSN virtual sensor. Accordingly, each of these properties is associated to a certain feature of interest, e.g. the air or water surface in some geographical region, an observed person moving in a defined area, etc. XGSN also considers the location of the virtual sensor, which is annotated with geo-location vocabularies. Finally, other specific metadata such as accuracy, operating range, and other capabilities can be added as we will see in the following subsections.

In the case of observations, XGSN considers that every tuple generated by a virtual sensor includes one observation per field, considering that every field corresponds to an observed property, in terms of the SSN Ontology. Nevertheless, the data from virtual sensors can range from low level measurements to complex events built on top of other virtual sensors and external data, as seen in Section 2. We provide a summary of the main ontology concepts used at the observation level by XGSN in Figure 4. While for low level observations (e.g. a particular NO_2 measurement at a certain point in time) XGSN can annotate values using the quantities ontology, for higher level concepts the observation may be symbolic and represent an alert or an actionable event. In the following

[7] For brevity, we represent ontology URIs in its prefixed form, e.g. ssn: denotes `http://purl.oclc.org/NET/ssnx/ssn#`.

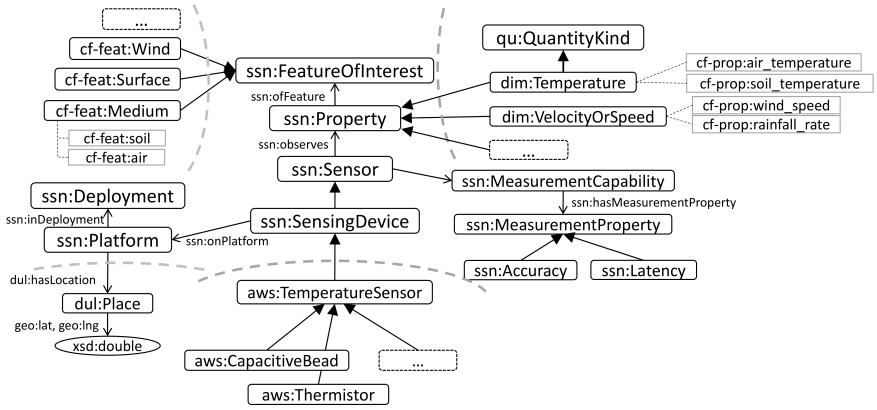


Fig. 3: Excerpt of some of the main ontology elements used by XGSN, based on the SSN ontology and QU/CF domain ontologies [17]. Notice the different extensions, marked by grayed dotted lines, with specialized ontologies defining specific sensors, features or observed properties.

sections we describe in more detail how this process occurs in XGSN, within the life-cycle of virtual sensors.

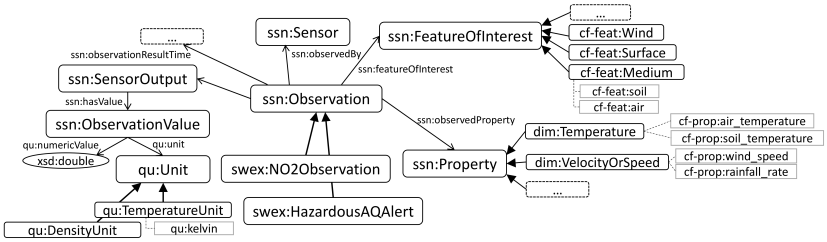


Fig. 4: Main ontology concepts used for observations in XGSN, based on the SSN ontology.

3.1 Virtual Sensor Registration

Virtual sensors in XGSN are set up using a configuration descriptor (an XML document as in Listing 1), and can be deployed and started at any time in an XGSN container. The output-structure element in the descriptor defines a set of fields for the virtual sensor. These fields are associated to observed properties according to the SSN ontology, and also correspond to the fields in the *query* element in the configuration. The wrapper information is also specified in this descriptor, and their parameters vary from wrapper to wrapper: e.g. the address of the data source, pull rates, security parameters, etc.

```
<virtual-sensor name="sens1" priority="10" >
  <processing-class>
    <class-name>org.openiot.gsn.vsensor.LSMExporter</class-name>
      ...
      <output-structure>
```

```
      <field name="temperature" type="double" />
      <field name="humidity" type="double" />
    </output-structure>
  </processing-class>
  <streams>
    <stream name="input1">
      <source alias="source1" sampling-rate="1" storage-size="1">
        <address wrapper="csv">
          ...
        </address>
        <query>select * from wrapper</query>
      </source>
      <query>select temp as temperature,humid as humidity, timed from source1</query>
    </stream>
  </streams>
</virtual-sensor>
```

Listing 1: Virtual sensor sample configuration in XGSN.

Each virtual sensor in a XGSN container has an associated sensor instance in an RDF cloud store (managed by the LSM middleware [16]) , i.e. a URI that uniquely identifies it. As we have seen in Figure 3, each sensor instance is connected to the respective properties, features of interest, location and other metadata needed by the system. All these metadata properties can be provided attached to the virtual sensor configuration, as in the example in Listing 2. In XGSN, we have limited the existing XML configuration solely to internal wrapper and processing class parameters, while all the high-level metadata of the sensors themselves is managed as RDF, and hence can be later shared as Linked Data. In the example, the sensor URI `http://openiot.eu/test/id/sensor/5010` observes air temperature, and has a number of other attributes including location, authorship, feature of interest, etc. Notice that the metadata can be extensible although XGSN internally requires only a handful of these, mainly the observed property, unit, feature and sensor type.

```
@base <http://openiot.eu/test/id/> .

<sensor/5010> rdf:type aws:CapacitiveBead,ssn:Sensor;
              rdfs:label "Sensor 5010";
              ssn:observes aws:air_temperature ;
              phenonet:hasSerialNumber <sensor/5010/serial/serial2> ;
              ssn:onPlatform <site/narrabri/Pweather> ;
              ssn:ofFeature <site/narrabri/sf/sf_narrabri> ;
              ssn:hasMeasurementProperty <sensor/5010/accuracy/acc_1> ;
              prov:wasGeneratedBy "AuthorName";
              DUL:hasLocation <place/location1>;
              lsm:hasSensorType <sensorType1>;

<sensor/5010/serial/serial2> rdf:type phenonet:SerialNumber;
                             phenonet:hasId "5010" .

<site/narrabri/Pweather>  rdf:type ssn:Platform ;
                          ssn:inDeployment <site/narrabri/deployment/2013> .
<site/narrabri/deployment/2013> rdf:type ssn:Deployment.

<sensor/5010/accuracy/acc_1> rdf:type ssn:Accuracy ;
                             qu:numericalValue "0.3"^^xsd:double ;
                             DUL:hasParameter phenonet:degreeCelsius .
```

Listing 2: Excerpt of a virtual sensor sample semantic descriptor in RDF used by XGSN. Prefixes ommitted.

The set up of the virtual sensor configuration and its annotation with the ontology constitutes the registration process, as illustrated in Figure 5. The registration and update of metadata can be performed using RESTful services provided by XGSN, simply by providing an RDF document with the required contents. In practice, the RDF metadata of virtual sensors is exposed as Linked Data by LSM, and can be queried or discovered by external applications and the upper layer components of the OpenIoT platform. Once the virtual sensor is registered and its metadata is available, it can produce (annotated) observations.

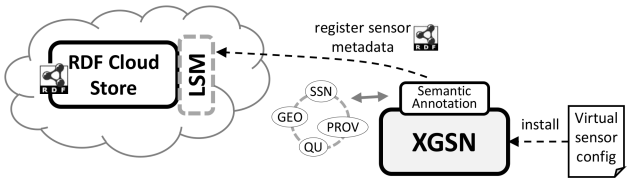


Fig. 5: Registration of a virtual sensor and annotation process performed in XGSN, storing the metadata through LSM.

3.2 Streaming Observation Annotations

Each time that the virtual sensor produces a value, XGSN annotates it and produces the corresponding observation according to the ontology model, as illustrated in Figure 6. Essentially, every time a tuple is produced in the virtual sensor stream (through the wrapper), a processing class automatically generates the RDF annotations that can be later transmitted to an RDF-aware data store or query processor. In the OpenIoT implementation, this processor is the LSM middleware, but it could even be processed by an RDF Stream Processor (RSP) such as CQELS [15], which is capable of evaluating continuous queries, extending the SPARQL language. Feeding any other continuous RDF query processor would follow a similar path: XGSN can feed the stream of RDF observations of an RSP. The advantage of this approach is that it decouples data acquisition from query processing, although it does add the complexity of having to manage both an RSP and XGSN. Also, RDF can be too verbose for certain stream processing tasks, depending on the volume and velocity of the data stream. Notice that XGSN observations could also be stored or processed through other channels, depending on the logic included in the virtual sensor processing class. For example, as it has been shown in [9], XGSN could expose a SPARQL-like (SPARQLStream) interface using R2RML mappings, through query rewriting techniques. Although for the OpenIoT reference implementation only the LSM integration has been wired, these types of extensions can be added in future stages.

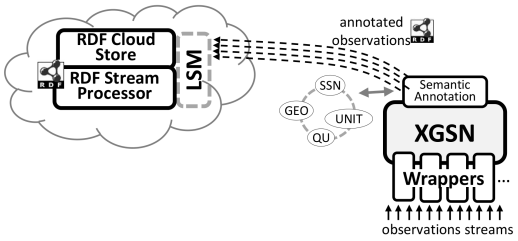


Fig. 6: XGSN annotation of observations produced by the virtual sensor, to the LSM middleware.

4 Architecture and Implementation

As already explained, the core abstraction in XGSN is the virtual sensor, which is hosted and deployed in a XGSN container. Each container is independent and runs its own set of virtual sensors, although different containers or instances may interchange data between them in a peer-to-peer fashion. Each container is structured in different layers, as detailed in Figure 7.

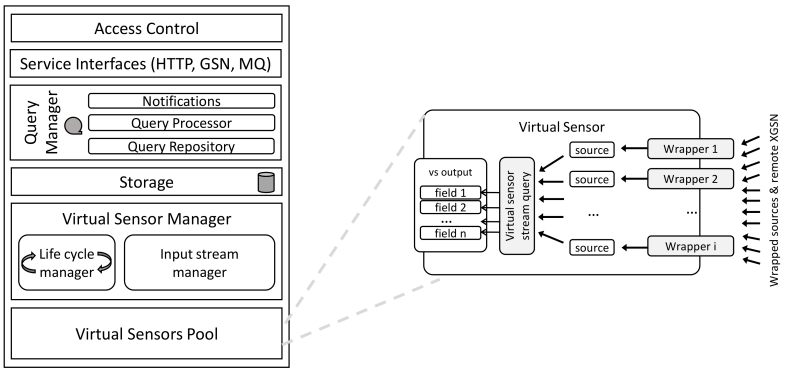


Fig. 7: XGSN container architecture, and virtual sensor acquisition and data stream provision [1].

A pool of deployed virtual sensors is administered by the virtual sensor manager. This includes handling the life-cycle of a virtual sensor (initialization, interactions, resources, disposal, etc.) and managing the incoming streams provided through the wrapper. The streams produced by each virtual sensor have an output structure, composed of one or more fields, which can be defined in terms of a continuous query that operates over one or more sources, each of them getting data through a wrapper. Notice that a wrapper can also encapsulate other (local or remote) virtual sensors, opening the possibility of having layered streams of data, as discussed in Section 2. Once the data is ready for processing, the storage layer handles persistent or temporary storage of the incoming data streams, depending on the virtual sensor configuration parameters. For some use cases

where observations need to be archived this may include storage in a relational database. Alternatively, in stream processing scenarios this can be handled in a memory-only database or a stream processor. Next, at the query manager layer, the system can host running queries that are continuously evaluated by a processor, acting directly on the streams produced at the lower layers. The query capabilities are exposed though the service interfaces, currently implemented as an HTTP RESTful interface that can be accessed by external applications. Moreover, each XGSN instance can be accessed through a native interface (inter-XGSN communication) implemented on top of ØMQ (ZeroMQ, see Section 5)[8]. Finally, there is an access layer on top of the services, that allows defining permissions over the virtual sensors and the observations they produce. More details about the internal architecture of XGSN can be found in [1].

The system has been implemented mainly in Java, while some out-of-the-box wrappers are implemented in other languages. The entire project is open-source, and is available in Github, as a standalone project[9] and also as part of the OpenIoT platform[10], with an existing and growing community of users and developers. The project documentation in the Github site provides more detailed information about the installation, deployment, development and production use of the system.

5 XGSN in a Distributed Environment

As we have mentioned, one of the main features of XGSN is its capability to work on a fully distributed mode, in such a way that data processing is as close as possible to the data sources. At the same time, this allows virtual sensors in one XGSN instance to be fed from other remote virtual sensors, enabling the definition of high level events that can be semantically annotated. We have experimented in a controlled environment how a network of XGSN instances works in this distributed scenario. We were interested in the generation rates, processing rates and network usage in our experimentation.

First, we used an XML-based protocol of exchange of observations between instances, and then we implemented an alternative and more efficient mechanism based in ZeroMQ. The first protocol is available in two versions: a push-based one that works in a publish-subscribe manner, and a pull-based that can work even if the client XGSN is behind a NAT (does not have a public IP address). The main advantage of this protocol is that is easier to debug, as it is human-readable, and that is based on well supported standards (XML and HTTP), but the overhead for processing the data distribution is not negligible. The alternative protocol, using ØMQ and the Java serialization library Kryo, is similar to the push version of the HTTP wrapper, using a PUB and SUB sockets, as shown on Figure 8. A proxy takes care of forwarding the subscriptions and the data to allow the simultaneous use of internal (*inproc* sockets) and external connections (by IP

[8] ZeroMQ: `http://www.zeromq.org/`
[9] GSN github repository: `https://github.com/LSIR/gsn/`
[10] OpenIoT github repository: `https://github.com/OpenIotOrg/openiot`

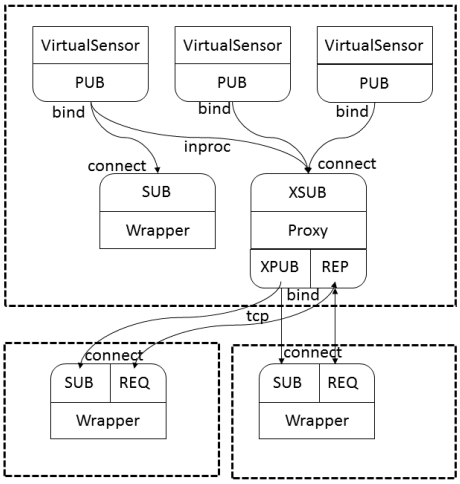


Fig. 8: XGSN ZeroMQ communication through asynchronous publish-subscribe sockets. Wrappers can subscribe to data of local or remote virtual sensors through a proxy.

address). It also serves as a directory, listing the available sensors and their data structure for external connections.

To evaluate the inter-server communication, we set up a use-case where each server is receiving or generating a stream of data and need to share it with all the other servers, in a kind of worst-case scenario. We deployed 24 XGSN servers, each on its own virtual machine, distributed over 9 physical machines. The virtual machines were provisioned with 2 cores (2.66GHz) and 3GB of RAM. Each server had 25 virtual sensors: one generating data every 10 ms and 24 connected to the other instances (including itself). The storage was kept in memory using the H2:mem database to reduce the disk writing overload.

In the first experiment, the remote HTTP XML wrapper was used to connect the 24 virtual sensors, in the second one the ZeroMQ wrapper was used with XML serialization and finally in the last experiment ZeroMQ with Kryo serialization. Each experiment lasted around 20 minutes during which two snapshots were taken.

In the first execution of the test-bed, using the remote wrapper and XML serialization, the CPU load of the virtual machines stayed around 36% and the network traffic was around 120kbps. The counter on the virtual sensor generating data indicated a rate of 90 element per seconds (Figure 9 (a)). This means XGSN needed 1 ms to generate the element and then wait for 10 ms before generating the next one. The network traffic is perfectly symmetric as every server is sending and receiving to, respectively from, all the others. From this results, it is clear that the network is saturated and cannot follow the element production. The second and third run, using the ZeroMQ wrappers presented a similar behavior regarding the CPU load and network traffic. CPU was almost at its maximum and network showed some differences in incoming versus outgoing

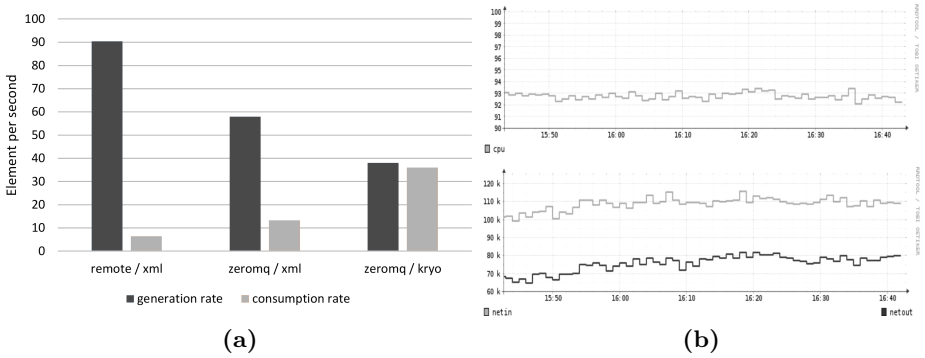


Fig. 9: Generation of data items in one XGSN instance (a); and CPU and network usage during the experiment with ZeroMQ (b).

traffic (Figure 9 (b)). This can be explained by the distribution of the generation rate among the servers. All XGSN instances received the same amount of data (same incoming traffic), but the ones generating less elements had also less data to send (lower outgoing traffic).

In the experiment using XML serialization, the communication protocol being lighter than HTTP, it was possible to send and process twice as much elements per seconds per virtual sensor (see Figure 10). But for processing those elements, the CPU was also more solicited (almost 100%) and was not able to keep the production rate. Finally using the Kryo serialization, the network was saturated, and similarly to the previous experiment the CPU had less time producing elements, around 38 per second in each instance. In this last experiment we almost reached the maximum performance possible with our virtual machines limitation: 860 elements sent, received and processed per second. In summary, we see that we can reach a fairly reasonable processing throughput, even more with the ZeroMQ implementation, although at the cost of losing reliability (relaxing packet loss guarantees).

6 Related Work & Discussion

Several systems have been devised to provide access to data streams on the Web in the form of Linked Data. Early approaches, including the architectures described in [18] and [13], rely on bulk-import operations that transform the sensor data into an RDF representation that can be queried using SPARQL in memory, lacking scalability and real-time querying capabilities. The Semantic Sensor Web [20] pioneered in bringing sensor data to the Linked Open data cloud, although it served more as a static repository without streaming or dynamic change in the observation data. Semantic annotations have also been considered at the service layer, for example for registering new sensors to observation services in [8]. In [12] an SOS service with semantic annotations on sensor data is

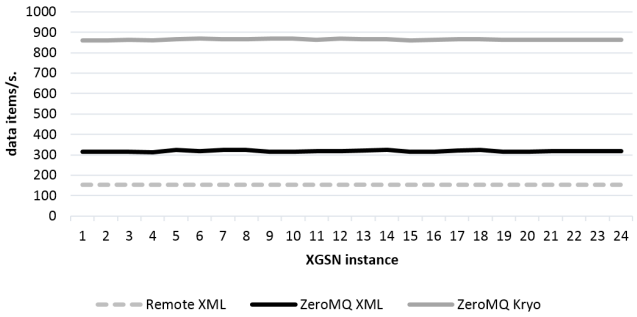


Fig. 10: Throughput in terms of data items received and processed per second, for the three configurations: Remote XML, ZeroMQ XML and ZeroMQ Kryo.

proposed, embedding terminology from an ontology in the XML of O&M and SensorML documents. In a different approach, the framework presented in [16] provides sensor data readings annotated with metadata from the Linked Data Cloud. This framework evolved into the LSM middleware that is now part of the OpenIoT platform, and that is used in conjunction with XGSN to manage the annotated sensor data and metadata. In most cases these systems have helped bringing sensor data to the (semantic) web, but resulted mainly in off-line archives of Linked Data, as opposed to the live annotation of sensor observations in XGSN. Moreover, we provide an end-to-end solution that manages the data from the acquisition layer up to the RDF and SPARQL data provision, through the LSM integration.

Other works have focused in the problem of continuous processing and querying over RDF streams, including CQELS [15], SPARQLStream [9], CSPARQL [5] or EP-SPARQL [2]. As explained in Section 3 these systems could be used in conjunction with XGSN, which can delegate the processing of the annotated observations to these systems, simply by implementing a processing class in a virtual sensor.

Nevertheless, even if our approach is capable of providing a solid semantic layer over IoT deployments, there are still many open challenges to tackle the problem of efficient stream processing. In the current OpenIoT implementation individual stream elements are annotated as they arrive, generating a non-negligeable volume of RDF which may be prohibitive for certain workloads. Stream compression techniques or virtualized RDF views over native data streams [9] are possible alternatives that have shown interesting results in other scenarios.

For handling continuous queries over streams, several Data Stream Management Systems (DSMS) have been designed and built in the past years, exploiting the power of continuous query languages and providing pull and push-based data access. Other systems, cataloged as complex event processors (CEP), emphasize on pattern matching in query processing and defining complex events from basic ones through a series of operators [11]. In recent years several commercial systems

have implemented the CEP paradigms, including Oracle CEP[11], StreamBase[12], Microsoft StreamInsight[13], IBM InfoSphereStream[14] and Esper[15]. Some of these systems provide similar (or alternative) streaming data techniques as those of XGSN, and they could even be used as an alternative processing class for XGSN virtual sensors. However, none of them provides semantically rich annotation capabilities on top of the query interfaces. XGSN could allow plugging different types of commercial CEPs, replacing its internal data streaming core, but the lack of query standards (such as SQL in the database world) makes it difficult to design such a mechanism.

Finally, there has been a large amount of work in the IoT community regarding suitable protocols for device-to-device and device-to-server communication. While XGSN is designed as a protocol-agnostic middleware (new protocols can be supported through new wrappers) it will be important in the immediate future to natively support these protocols. For this it is also envisaged to allow deploying a constrained version of XGSN inside sensors and mobile devices, so that these *things* can transparently communicate with standard XGSN instances and therefore with the Web.

7 Conclusions

We have presented XGSN, an open-source middleware that is capable of collecting data from sensors and things in the real world, abstracted as virtual sensors, process them and publish the data using a semantic model based on the SSN ontology. We have shown in detail how the annotation process has been designed and implemented, for both the sensor metadata and the produced observations. We have also described a multi-layered scheme for defining observations at different levels of abstraction, for which XGSN provides a very flexible, extendable and scalable infrastructure. We have shown how the system goes beyond other existing sensor middleware, by adding the semantic aspect at its core, and by integrating its existing features and complementing them with the LSM framework for RDF storage and querying. XGSN is a fully functional and ready-to-use system, with a growing community of users and developers, and is now part of the wider OpenIoT platform. XGSN has been shown to be effective and useful in several different types of use cases, including air quality monitoring, environmental alpine experimentation, participatory sensing, smart agriculture, intelligent manufacturing, etc.

We plan to add several features to XGSN in the near future. First, we plan to make use of the semantic annotations of virtual sensors to allow enhancing the M2M communication among XGSN instances or even other *semantics-aware* applications. We also plan to integrate the semantic features not only with LSM

[11] Oracle:`http://www.oracle.com/technetwork/middleware/complex-event-processing/overview`
[12] StreamBase: `http://www.streambase.com/`
[13] StreamInsight: `http://msdn.microsoft.com/en-us/sqlserver/ee476990`
[14] InfoSphereStream: `http://www-01.ibm.com/software/data/infosphere/streams/`
[15] Esper: `http://esper.codehaus.org/`

but with other backends (not only cloud-based but also local-based). Another future direction is exploring parallelized execution of streaming data algorithms over the observation data (e.g. Spark[16] or Storm[17]), and how these can be combined with our system. There is also room for work integrating this system with mobile sensing and participatory sensing, where the mixture of incentives and privacy can be a challenging problem. While there is a need for having accurate data from a crowdsensing community, it is also important to protect privacy of individuals contributing to the dataset. Finally, we are re-designing the web services interfaces of XGSN, expanding its functionalities (e.g. including discovery, exploiting the semantic annotations for linkage, provenance support, etc.) and adhering to the Linked Data Platform.

Acknowledgments Partially supported by the OpenIoT FP7-287305 and the SNSF-funded Nano-Tera OpenSense2 projects.

References

1. Aberer, K., Hauswirth, M., Salehi, A.: A middleware for fast and flexible sensor network deployment. In: Proc. 32nd International Conference on Very Large Data Bases VLDB, pp. 1199–1202. VLDB Endowment (2006)
2. Anicic, D., Fodor, P., Rudolph, S., Stojanovic, N.: EP-SPARQL: a unified language for event processing and stream reasoning. In: Proc. 20th International Conference on World Wide Web, pp. 635–644 (2011)
3. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer networks 54(15), 2787–2805 (2010)
4. Bandyopadhyay, S., Sengupta, M., Maiti, S., Dutta, S.: Role of middleware for internet of things: A study. International Journal of Computer Science and Engineering Survey 2(3), 94–105 (2011)
5. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.: Incremental reasoning on streams and rich background knowledge. In: Proc. 7th Extended Semantic Web Conference, pp. 1–15 (2010)
6. Barnaghi, P., Wang, W., Henson, C., Taylor, K.: Semantics for the internet of things: early progress and back to the future. International Journal on Semantic Web and Information Systems (IJSWIS) 8(1), 1–21 (2012)
7. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. International Journal on Semantic Web and Information Systems 5(3), 1–22 (2009)
8. Bröring, A., Janowicz, K., Stasch, C., Kuhn, W.: Semantic challenges for sensor plug and play. In: Proc. 9th International Symposium on Web and Wireless Geographical Information Systems. vol. 5886, pp. 72–86. Springer (2009)
9. Calbimonte, J.P., Jeung, H., Corcho, O., Aberer, K.: Enabling query technologies for the semantic sensor web. International Journal On Semantic Web and Information Systems (IJSWIS) 8(1), 43–63 (2012)
10. Compton, M., Barnaghi, P., Bermudez, L., García-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W.D., Phuoc, D.L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page,

[16] https://spark.apache.org/streaming/

[17] http://storm.incubator.apache.org/

K., Passant, A., Sheth, A., Taylor, K.: The SSN ontology of the W3C semantic sensor network incubator group. Journal of Web Semantics 17, 25–32 (2012)
11. Cugola, G., Margara, A.: Processing flows of information: From data stream to complex event processing. ACM Computing Surveys 44(3), 15:1–15:62 (2011)
12. Henson, C., Pschorr, J., Sheth, A., Thirunarayan, K.: SemSOS: Semantic Sensor Observation Service. In: Proc. International Symposium on Collaborative Technologies and Systems CTS 2009. pp. 44–53. IEEE (2009)
13. Huang, V., Javed, M.: Semantic sensor information description and processing. In: Proc. 2nd International Conference on Sensor Technologies and Applications SENSORCOMM 2008. pp. 456–461. IEEE (2008)
14. Janowicz, K., Scheider, S., Pehle, T., Hart, G.: Geospatial semantics and linked spatiotemporal data–past, present, and future. Semantic Web 3(4), 321–332 (2012)
15. Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.: A native and adaptive approach for unified processing of linked streams and linked data. In: Proc. 10th International Semantic Web Conference ISWC, pp. 370–388. Springer (2011)
16. Le-Phuoc, D., Nguyen-Mau, H.Q., Parreira, J.X., Hauswirth, M.: A middleware framework for scalable management of linked streams. Web Semantics: Science, Services and Agents on the World Wide Web 16, 42–51 (2012)
17. Lefort, L., Henson, C., Taylor, K.: Semantic sensor network xg final report. Tech. rep., W3C SSN XG (2011), http://www.w3.org/2005/Incubator/ssn/XGR-ssn/
18. Lewis, M., Cameron, D., Xie, S., Arpinar, B.: ES3N: A semantic approach to data management in sensor networks. In: Proc. 1st International Workshop on Semantic Sensor Networks SSN 2006 (2006)
19. Pfisterer, D., Romer, K., Bimschas, D., Kleine, O., Mietz, R., Truong, C., Hasemann, H., Kroller, A., Pagel, M., Hauswirth, M., et al.: Spitfire: toward a semantic web of things. Communications Magazine, IEEE 49(11), 40–48 (2011)
20. Sheth, A., Henson, C., Sahoo, S.S.: Semantic sensor web. Internet Computing, IEEE 12(4), 78–83 (2008)
21. Song, Z., Cárdenas, A.A., Masuoka, R.: Semantic middleware for the internet of things. In: Internet of Things (IOT), 2010. pp. 1–8. IEEE (2010)

Sensor Data Provenance: SSNO and PROV-O Together at Last

Michael Compton[1], David Corsar[2], and Kerry Taylor[1,3]

[1] CSIRO Digital Productivity and Services, Canberra
firstname.lastname@csiro.au
[2] University of Aberdeen, Aberdeen, UK dcorsar@abdn.ac.uk
[3] Australian National University

Abstract. This paper presents an alignment between the W3C Provenance Working Group's recommended ontology (PROV-O) and the W3C Semantic Sensor Networks Incubator Group's ontology (SSNO). The alignment views PROV-O as an upper ontology which is extended with SSNO concepts and properties. This allows representation of observation details and sensor deployments that are not possible in the SSNO alone, and gives a basis for alignment with Open Geospatial Consortium Observations & Measurements aligned ontologies. Further to the alignment, rules are presented that further constrain the interpretation of the aligned ontologies and provide a mechanism by which provenance information can be generated from SSN data thereby allowing modellers to take advantage of the new features. The benefit of the aligned ontologies is illustrated with an example of cross-domain provenance querying enabled by the alignment.

1 Introduction

Sensor deployments for smart cities, the Internet of Things, crowd sensing, and environmental research produce large volumes of data, which, when analysed either in real-time or using archives, have the potential to impact on virtually all aspects of society [2]. The Semantic Sensor Web can play an important role in realising these impacts by supporting the identification, selection and use of sensor data through expressive representations of sensor resources. This support can be extended by providing additional details, such as how the data was produced, any processing to which it was subjected, and who was involved in those steps, i.e. the data's provenance [2, 21]. Provenance is valuable here as it can be used to assist users with understanding, verifying, and assessing the data's quality and trustworthiness before use [20, 28, 17].

Due to the work of the W3C Semantic Sensor Networks (SSN) Incubator Group and W3C Provenance Working Group, the semantic web community now has OWL2 [29] ontologies designed to support the reuse and interoperability of both sensor and provenance information. The SSN ontology (SSNO) [5, 16] has been adopted as a defacto standard for the semantic specification of sensors, sensor devices, systems, processes, and observations. The SSNO is designed to

capture both historical details (for example, details of deployments and how observation values were produced) along with current information (for example, sensor capabilities). The PROV-O ontology [14] is a W3C recommendation for the representation and interchange of provenance information on the Web, where provenance is defined as information about the entities, activities, and people involved in the production of things - data, physical objects, etc. [20]. PROV-O builds on significant research by the database, workflow, and e-science communities (see [3, 19, 25] for relevant reviews).

This paper contributes an alignment between the SSNO and PROV-O models, including rules that both constrain the interpretation of the alignment and can be used for inferring provenance from sensor data. The alignment extends the expressive capability of the SSNO for recording observations and system deployments, enabling more comprehensive historical information to be described than is possible using SSNO alone. As such, the alignment also serves as a best practice guide for using SSNO and PROVO to represent the provenance of observations, observation values, and sensor system deployments. The alignment also enables the integration of SSNO data with other data expressed using PROV-O, and so makes SSNO data available to tools and applications capable of consuming PROV-O (e.g. for visualisations).

Paper Outline: First, the SSN (§2.1) and PROV (§2.2) ontologies are discussed. Then related work (§3) is reviewed. The alignment (§4) is then presented in terms of a central pattern (§4.1), aligning the Stimulus-Sensor-Observation pattern from the SSNO to PROV-O along with an alignment (§4.2) for the SSNO's platforms and deployments model. Next, rules are given (§4.3) that further inform the alignment and can be used to produce data in the aligned ontology from SSNO data. An example (§5) illustrates the use of the alignment. The paper concludes (§6) with a discussion of the alignment and its features.

2 Background

The PROV and SSN ontologies are accessible from their respective namespaces:

`http://www.w3.org/ns/prov`, and
`http://purl.oclc.org/NET/ssnx/ssn`

Throughout, the namespaces for the PROV and SSN ontologies are abbreviated as 'prov' and 'ssn' respectively. Hence ssn:Sensor means the concept `http://purl.oclc.org/NET/ssnx/ssn#Sensor`. The SSN ontology uses the DOLCE Ultralite ontology, called DUL, as an upper ontology and its namespace, `http://www.loa-cnr.it/ontologies/DUL.owl`, is abbreviated as 'dul'. Concepts and properties given without a namespace are those of the alignment discussed here.

2.1 SSN

The SSN ontology was designed to describe sensors: what is observed, how observations are made, the observations, and the qualities of the sensors and ob-

servations. Full descriptions of the SSNO are given in Compton et al. [5] and the incubator group's final report [16].

The SSNO is built around the Stimulus-Sensor-Observation pattern [11] that describes the relationship between an observing ssn:Sensor, the ssn:Property measured, the real-world ssn:Stimulus, the ssn:Sensing procedure followed and the resultant ssn:Observation.

The SSNO expands on the central pattern to describe the ssn:Accuracy, ssn:Frequency, ssn:Drift, etc. of a sensor as its ssn:MeasurementCapability, to describe the ssn:OperatingRange and ssn:SurvivalRange of sensors, and to provide a skeleton structure for describing how a sensor may be attached to an ssn:Platform and used in an ssn:Deployment.

However, it is the Stimulus-Sensor-Observation pattern that forms the key part for the alignment to PROV-O.

2.2 PROV

The PROV-O recommendation is an OWL2 encoding of the PROV Data Model [20], which describes provenance in terms of relationships between three main types of concepts: prov:Entity, which represents (physical, digital, or other types of) things; prov:Activity, which occur over time and can use and/or generate entities; and prov:Agent, which are responsible for activities occurring, entities existing, or another agent's activity [14].

Relationships between these concepts describe the influence one has had on another. These include that an activity prov:used and prov:generated entities, ascribing an entity to an agent (prov:wasAttributedTo), and an agent to an activity (prov:wasAssociatedWith). The nature of the influence can be defined using qualified relations to describe the prov:Role of the entity, agent, or activity. Qualified relations include: prov:Usage, which defines the role of an entity used by an activity; and prov:Association, which defines the role of an agent in an activity, along with any prov:Plan the agent was following during the activity.

3 Related Work

The provenance of sensor data has many uses, including: supporting the understanding and reuse of sensor data, including data that has been aggregated or otherwise processed [21]; ensuring the correct attribution of publicly available sensor data [4]; supporting users trace the involvement of sensor data in experiments for reproducibility purposes [10]; searching for, and identifying sensor data within data stores [15]; verifying data transmitted through nodes in a sensor network [24]; and supporting quality [6] and trustworthiness assessments [28]. Despite this, there are few published alignments between sensor and provenance ontologies, which are discussed below.

The Open Provenance Model (OPM) [18] is used by Lie et. al. [17] to record the provenance of virtual sensors within the Tupelo semantic content management system. OPM integrates the sensor registration and selection events with

the retrieval of raw data and model-based transformations applied to derive new data. In this system, data is modelled as the OPM equivalent of prov:Entity and actions as prov:Activity; no further alignments are described.

Patni et. al. [22] use the Provenir ontology [23] to capture and store the provenance of sensor data in their sensor management system. Alignments between Provenir and their sensor ontology are defined through a series of rdfs:subClassOf and rdfs:subPropertyOf relationships modelling sensor-specific provenance information. This includes modelling observation values as the Provenir equivalent of prov:Entity and the sampling time property as the equivalent of prov:atTime.

Stasch et. al. [26] describe their extension of the SSNO to represent aggregations of observations, and detail the use of the Provenance Vocabulary[4] and OPM to record each aggregation's provenance. Observations and aggregations are modelled using the Provenance Vocabulary and OPM equivalents of prov:Entity, with aggregations being created by an aggregation activity that used observations. However, it is unclear if the remaining SSNO concepts have been aligned to a provenance model, or if these alignments have been explicitly defined in an ontology to enable their reuse.

In the context of a citizen sensing application, Corsar et. al. [6] define a partial alignment between the SSNO and PROV-O. The alignment is restricted to defining subclass relationships between the SSNO observation, sensor, and sensing concepts and the PROV-O entity, agent, and activity concepts respectively. The alignment is used to integrate SSNO data with other data via the PROV-O model to support quality assessment of observations from citizen sensors.

These various works illustrate a requirement to combine sensor and provenance models, and potential uses of the resulting model. However, to the best of our knowledge, there does not currently exist a comprehensive alignment between any established sensor and provenance models. We have therefore chosen to develop such an alignment between the two ontologies (SSNO and PROV-O) that are now accepted within (and beyond) their respective communities as the main reference models.

4 SSNO-PROV-O Alignment

While the SSN and PROV ontologies are compatible in some areas, the two are modelled from different perspectives. Largely, the SSN ontology is about properties and potential: what sensors measure, how they measure, and the qualities of such measurements. While, on the other hand, the PROV ontology models what has occurred and how things were made: what the entities are, what produced them and how. The potential for overlap and alignment between the two ontologies is observations. Observations are the things that are produced by sensors and, thus from a provenance perspective, this production or generation

[4] `http://trdf.sourceforge.net/provenance/ns.html`

is the key point in linking the two ontologies. Indeed, it is observations around which the following alignment is built.[5]

The PROV ontology is the more abstract of the two and thus the alignment places SSNO concepts and relations into the PROV-O hierarchy, making them subconcepts and subproperties of PROV-O concepts and properties. The PROV-O ontology is used like an upper ontology in this respect. This approach allows, for example, other provenance data and modelling to be used in conjunction with the sensor data in a provenance setting.

New concepts extending the PROV-O hierarchy are created to further glue the two ontologies together, as simply placing SSNO concepts into the PROV-O hierarchy does not complete the full richness of the alignment.

In making the alignment, modelling choices are present even at the initial stages and each choice has far reaching consequences for how other aspects are aligned. The placement of ssn:Observation and ssn:Sensor are the most central.

A choice was made in the SSN ontology to make an ssn:Observation a dul:Situation (i.e.: ssn:Observation $\sqsubseteq$ dul:Situation). That is, an observation is an interpretation of real-world events and the results of those events: for example, a stimulus (wind) spins the cups on a wind sensor, generating a current, and through an equation modelling the relationship between this and the physical property of wind speed the sensor outputs a value that we can choose to interpret as an observation of the wind speed at that moment. The observation is the social construct of the interpretation, not the act of the observing itself.

On the other hand, the Open Geospatial Consortium (OGC) Observations and Measurements (O&M) (previously an OGC standard [7, 8], now an ISO standard [1]), sees an observation as an event: the event of sensing and producing the result. The SSNO and O&M attach essentially the same data to an observation — a value, a feature of interest, an observed property, etc. — but place the observation itself in a different context. The SSNO argues that O&M conflates two aspects of the observation: the act and the interpretation.

This dichotomy poses an immediate choice in the alignment. Following the SSNO model, ssn:Observation would be aligned with prov:Entity (ssn:Observation $\sqsubseteq$ prov:Entity). But that choice reinforces the distinction between O&M and the SSNO. It may also make the alignment less useful as it would not fit with OGC models and would not, for example, be able to represent data from an O&M aligned ontology, such as that given by Cox [9]. Aligning to O&M would align ssn:Observation to prov:Activity (ssn:Observation $\sqsubseteq$ prov:Activity). Such a choice might be passable in an SSN Ontology not aligned to DOLCE, but with the DOLCE alignment it would be problematic as an observation would be both a dul:Object and a prov:Activity. Since dul:Object is disjoint from dul:Event but PROV-O specifies no disjoints this alignment may not lead to inconsistency, but would be ontologically uncomfortable.

Instead of following either approach, the alignment reconciles these disparate viewpoints. It aligns the SSNO approach to PROV-O and describes new PROV-

[5] The alignment ontology is available at `http://purl.oclc.org/NET/ssnprov/ssnprov`.

O aligned concepts for the O&M approach, linking them through provenance. That is, the ssn:Observation is reached by a prov:Activity that interprets the act of sensing. The rest of the alignment follows from this central pattern.

Figure 1 shows the central pattern of the alignment, while Figure 2 shows the full alignment.

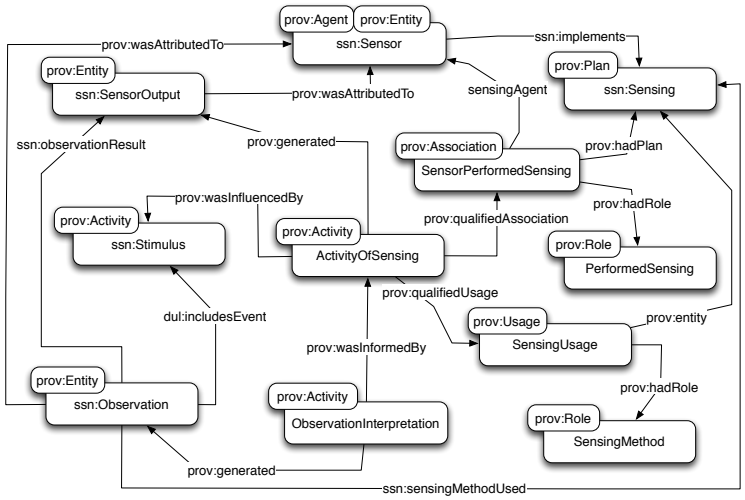


Fig. 1. The central pattern of the alignment. Boxes represent concepts, with inheritance represented by the smaller boxes. Arrows represent restrictions on concepts of the form $\exists r.C$. The figure focuses on the relationships that can be specified and inferred as they that are the key new features of the alignment, but inheritance relationships are also shown.

4.1 Alignment Pattern

The SSNO concept of observations, ssn:Observation, is classified as a prov:Entity (ssn:Observation $\sqsubseteq$ prov:Entity). A prov:Entity being "[...] a physical, digital, conceptual, or other kind of thing with some fixed aspects" [14], ssn:Observation, social situations or constructs, are thus a conceptual thing.

Sensors, ssn:Sensor, are classified as both prov:Entity and prov:Agent (i.e. both ssn:Sensor $\sqsubseteq$ prov:Entity and ssn:Sensor $\sqsubseteq$ prov:Agent). prov:Entity because sensors are physical or digital things in the sense meant by the PROV-O definition above; and prov:Agent because in performing the act of sensing they are the agents that enact a prov:Activity and because sensors are responsible for the existence of observations. An agent in PROV-O being "[...] something that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent's activity" [14].

Fig. 2. The alignment of the SSNO to PROV-O. Grey indicates assertions from the SSNO provided here for reference.

Sensors and observations are linked in the SSNO by ssn:madeObservation and ssn:observedBy. The alignment adds prov:wasAttributedTo, i.e. that the observation entity was attributed to the sensor agent. Further linking the two are new concepts SensorPerformedSensing, ActivityOfSensing and ObservationInterpretation, which add the detail that describe the observation activity and fill in the O&M perspective as discussed in above.

The new concept ActivityOfSensing is the prov:Activity of performing the sensing. An ActivityOfSensing prov:generated the ssn:SensorOutput, was influenced by (prov:wasInfluencedBy) the ssn:Stimulus and, through SensorPerformedSensing (a prov:Association), prov:wasAssociatedWith the sensor. Further, an ActivityOfSensing may be specified as prov:wasInformedBy things such as the feature of interest or the observed property, though this isn't required in the alignment. It is the ActivityOfSensing that fills in the O&M perspective of observation.

The prov:Activity of ObservationInterpretation records the activity that interpreted the results of an ActivityOfSensing and resulted in (prov:generated) a dul:Situation that is the ssn:Observation. The activity of ObservationInterpretation prov:wasInformedBy the ActivityOfSensing. As with the ActivityOfSensing, the ObservationInterpretation may be prov:wasInformedBy some of the aspects recorded by the ssn:Observation.

Observations themselves can record in SSNO the ssn:Stimulus as a related event (dul:includesEvent) as well as a feature of interest and observed property. Here, as with ActivityOfSensing and ObservationInterpretation, the alignment is left open. These links could be added by specifying them as prov:wasInfluencedBy or prov:wasInformedBy, but we chose to leave the alignment flexible here and allow such links to be included, but not mandate them.

Specifying ssn:observationResult $\sqsubseteq$ prov:wasDerivedFrom shows the provenience attribution of an observation as a situation being partly derived from the observation result.

The ssn:SensorOutput is prov:wasAttributedTo the sensor agent by virtue of specifying ssn:isProducedBy $\sqsubseteq$ prov:wasAttributedTo. Some properties, of which ssn:isProducedBy is one, were not found to have alignments to DUL in the development of the SSNO. This is remedied in the alignment, which takes advantage of PROV-O as the upper ontology to further restrict the interpretation of ssn:isProducedBy.

The alignment adds further nuances to the description of an observation that SSN cannot do alone. SensorPerformedSensing can be enriched to show that the sensor had the role of performing the sensing in the ActivityOfSensing (prov:hadRole and PerformedSensing). More importantly, it can show the plan (prov:hadPlan) that was used to perform the sensing — linking the activity of sensing with the ssn:Sensing plan that was used. The SSNO alone can show what sensing plans a sensor is capable of performing, but, for an individual observation, the SSNO cannot show which plan was enacted. This advantage in the SSN-PROV-O alignment is helpful in specifications of the observations of multi-instruments and systems with complex sensing options, where the provenance can now record more accurately what was done.

The dul:Plan of ssn:Sensing in the SSN ontology can also be used to express if the observation was made in some particular way: i.e. specifying that a sensor ssn:implements some ssn:Sensing describes how the sensor works, while specifying that an observation had a ssn:sensingMethodUsed of some ssn:Sensing describes how the sensor was used in making the observation — a particular configuration or physical setup for example. The alignment enriches the picture by showing that it is the activity of sensing that used the plan. For this the alignment shows that an ActivityOfSensing can have a prov:qualifiedUsage of some SensingUsage.

The alignment further ensures that the dul:Plan a ssn:Sensor enacts in an ActivityOfSensing must be a plan that it ssn:implements by stating the role chain:

$$\text{sensingAgent}^{-} \circ \text{prov:hadPlan} \sqsubseteq \text{ssn:implements}.$$

There is no alignment to PROV-O for aspects of SSNO such as the measuring properties (ssn:MeasurementCapability) or ssn:OperatingRange as these are the static aspects of sensors that are covered by the SSNO and not PROV-O. That is, because aspects such as accuracy and drift, or specifications of operating and survival ranges are inherent properties of sensors, they have no natural alignment into PROV-O.

In fact, there is no alignment to PROV-O for any ssn:Property as properties such as accuracy (which is ssn:MeasurementProperty) or survival temperature (which would be ssn:SurvialProperty) being measurable properties of sensors, just as temperature is a measurable property of a location. Properties could be aligned as prov:Entity (a conceptual entity in PROV-O), but since properties are inherent for an object and PROV-O doesn't provide mechanisms for talking about and linking entities, except through creation or participation in activities or membership of collections, there is no useful way to align properties to PROV-O. For example, one might want to describe the provenance of a sensor, including its creation and creator, but such a description is unlikely to involve specifying that the creator made or generated the accuracy of the sensor.

That there are aspects of the SSNO not covered by PROV-O and parts of the alignment that provide extra capability to the SSNO shows that the alignment extends each ontology.

A further alignment is possible for the deployments and platforms aspects of the SSNO as this describes time-varying aspects of sensors. Again the alignment adds expressive power not available in SSN alone.

4.2 Deployments Alignment

In the SSNO deployments are DUL processes (ssn:Deployment $\sqsubseteq$ dul:Process): that is, events about which the evolving nature is important. Such a conception of a deployment as representing the ongoing process of initial deployment, maintenance, addition and removal, recalibration, etc. fits naturally with PROV-O. The alignment, however, adds to the expressive capability of the SSNO, and more nuanced and clear specifications can be made in the alignment, giving the expressive capability to state subtle properties of the evolution and nature of deployments and thus may give further reasoning power for sensor search and selection.

The alignment for deployments (bottom of Figure 2) makes the assertions that the processes for deployment are prov:Activity, that both ssn:Platform and ssn:System are prov:Entity, and that an ssn:System is a prov:Collection. Together with the properties assertions, the whole of the SSNO deployments skeleton is aligned to PROV-O.

Consider, for example, Figure 3 which shows an SSNO description of a deployment (black) augmented with a description that can be achieved in the aligned ontologies (grey). The SSNO can describe the deployment and the processes, systems and platforms involved, but can't show the relationships and derivations between the parts. Once the PROV-O parts are added, the usage, generation and temporal dependencies are clear, though one may wish to further specify these with timing assertions on the activities, such as with prov:startedAtTime and prov:endedAtTime, or by using dul:precedes and dul:follows.

If desired, information about the agents associated with the deployment-related processes can also be defined (not expressible in the SSNO). This, in turn, can include details explaining why the action took place: for example, as part of the system's maintenance plan or in response to the system malfunctioning.

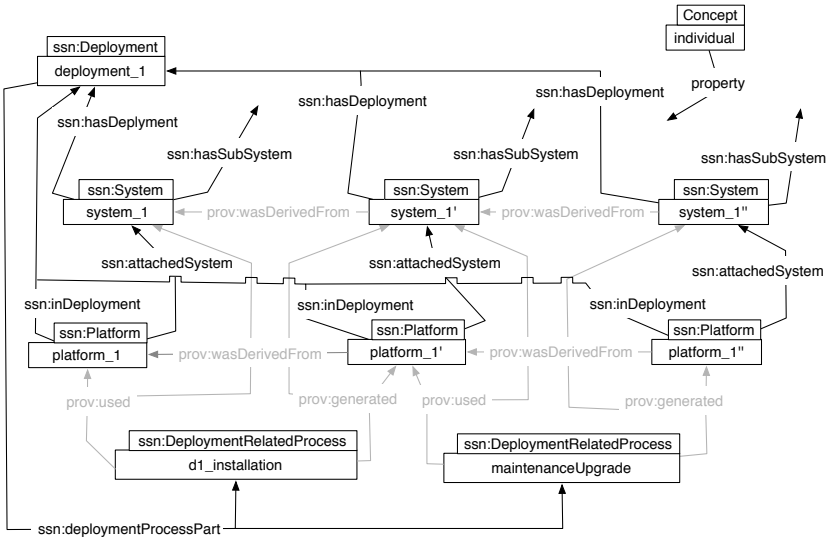


Fig. 3. An example of deployments. Black indicates SSNO only modelling; grey the extra modelling in the aligned ontology.

Further subtle nuances can be gained, for example, by adding that system_1', system_1", platform_1' and platform_1" from Figure 3 are revisions of system_1 and system_1. This could be done in the aligned ontology by specifying prov:wasRevisionOf. A more complete specification could be gained by specifying temporal parts using a 4D fluents [30] approach.

Such detailed modelling could be used for example when sensor selection is dependent on dynamic properties of sensors, such as consistency of maintenance or time since last calibration. This information can also be used for identifying and selecting sensor data or to make judgements about the data from providers or networks based on the maintenance history of other deployments made by the provider. Further, calibration and maintenance events could be used, for example, in conjunction with a sensor's static specification of accuracy and drift to determine the likely current performance of the sensor based on its history. This, in turn, can be used during, for example, quality assessments of observations. Information about the agents involved in the deployment can be beneficial when making trust assessments [28], or for data access control (for example, restricting access to only members of the organisation responsible for the sensor).

4.3 Rules

As is often the case, the OWL specification restricts our interpretation to a set of potential satisfying models, but one may wish to add further detail. For this we

have defined a set of rules[6] to further constrain the interpretation of the model and which can be used to generate aligned provenance data from SSN data.

The rules define how the concepts introduced by the alignment should be interpreted in the context of SSN data. Two types of rules could be considered: the first define ObservationInterpretation and ActivityOfSensing; the second define the relationships, particularly from ActivityOfSensing to the remaining concepts.

Implementations of these rules have been produced using the Stardog 2.1.3[7] and SPARQL Inferencing Notation[8] (SPIN) formats. The following shows fragments of our Stardog rules. Stardog rules[9] are written in a SPARQL-like syntax with an 'if ... then ...' structure. The Stardog rules are activated on query and, although these infer the existence of new individuals and can use the inferred individuals in answering queries, the inferred individuals are not persistent. The SPIN rules, however, can persist the inferred individuals, but the SPIN reasoner is limited to using Jena[10] compatible models.

The following fragment of a Stardog rule infers an ObservationInterpretation based on the existence of an ssn:Observation.[11]

```
IF {
   ?obs rdf:type ssn:Observation .
   FILTER NOT EXISTS (?obsI prov:generated ?obs .
      ?obsI rdf:type prov_ssn:ObservationInterpretation) .
   BIND (UUID() AS ?obsInterp).
} THEN { ?obsInterp prov:generated ?obs .
   ?obsInterp rdf:type prov_ssn:ObservationInterpretation.
}
```

The following example rule shows how the SensingUsage qualified association can be derived for an ActivityOfSensing.

```
IF {
   ?obs rdf:type ssn:Observation .
   ?obs ssn:sensingMethodUsed ?sensing .
   ?obsI prov:generated ?obs .
   ?obsI rdf:type prov_ssn:ObservationInterpretation .
   ?obsI prov:wasInformedBy ?actOfSen .
   ?actOfSen rdf:type prov_ssn:ActivityOfSensing.
   FILTER NOT EXISTS (?actOfSen prov:qualifiedUsage ?senUse .
      ?senUse rdf:type prov_ssn:SensingUsage) .
   BIND (UUID() AS ?sensingUsage).
} THEN { ?actOfSen prov:qualifiedUsage ?sensingUsage .
   ?sensingUsage rdf:type prov_ssn:SensingUsage .
```

[6] The rules are available at `http://purl.oclc.org/NET/ssnprov/rules`.
[7] `http://stardog.com/`
[8] `http://spinrdf.org/`
[9] `http://docs.stardog.com/owl2/`
[10] `https://jena.apache.org/`
[11] The UUID() function creates a unique identifier for the new individual.

```
    ?sensingUsage prov:entity ?sensing .
    ?sensingUsage prov:hadRole prov_ssn:SensingMethod.
}
```

5 Example

One of our motivations for this paper is to support the enrichment of PROV-O with sensor-specific concepts of relevance to provenance. Simultaneously, we enable PROV-O to act as a common language for modelling provenance-like interactions between the data produced by sensors and non-sensor data, such as that sourced from social networks or simulation systems. In this example we validate those aims by demonstrating the interaction of our alignment with the conceptual model *CERIF 1.3*. We show queries over provenance independently recorded as SSNO or CERIF but jointly queryable (a) through PROV-O by virtue of the alignment mappings and basic OWL inference and (b) through the extended terminology of both SSNO and CERIF for domain-specific precision.

CERIF 1.3 is natively described as a large relational data model that supports the management of research information, associating comprehensive information about European research projects and infrastructure with their resources and products [12]. The model includes sophisticated support for encoding ontological relationships between the concepts generally represented as tables in the model. A comprehensive re-interpretation of the model as an OWL ontology could both unify the "semantics" represented in the relationships with the semantics represented as relational attributes, and could also improve the interoperability of data using linked data approaches. An initial OWL re-interpretation described in [13] models a few of the relationships but does not cover any attributes.

CERIF describes concepts that could be mapped to PROV-O and thereby enrich the provenance of research results, being patents, publications or products such as datasets and software. CERIF links these results to equipment, funding, people, and impact indicators. We have developed a partial mapping to PROV-O based on our own partial encoding of CERIF as an OWL 2 ontology, which enables us to demonstrate cross-cutting SPARQL queries that relate information arising from the domain of SSNO to information in the domain of CERIF, using PROV-O as a lingua-franca while using each of those ontologies for greater specificity when required.

For the queries, let us assume a scenario where we have many sensors installed on an experimental farm that is used as a research facility by many independent research projects. The sensor network is described by SSNO, so the provenance of research results can be tracked back to data sources. In fact, this scenario has been realized [27]. Let us further assume that CERIF (with namespace prefix 'cf') is used for those research projects, and that the mapping described above is deployed. Now, a paddock inspection reveals that a post, kirbypost23 on which several sensing devices were mounted, has fallen down. At least some of the sensors appear to be continuing to operate, but they would be unreliable, especially the automatic weather station wxt520 that needs to be

both vertical, unobstructed and at a known elevation above the ground for wind and precipitation measurements. We want to contact the principal investigators of research projects that are using those sensors to let them know that the data might be of poor quality.

In the first query we use PROV-O terms exclusively to retrieve individuals that are encoded as instances of either SSNO or CERIF classes, but which are related through PROV-O by virtue of the respective alignments. By this we demonstrate the lingua-franca value of the alignment. The query retrieves all projects (activities) and their associated people (agents) that either used the weather station or used the sensor network that included the weather station.

```
SELECT ?proj ?sys ?pers
    WHERE {
        {  {?proj prov:used ?sys . ?sys prov:hadMember :wxt520}
        UNION
           {?proj prov:used :wxt520}
        }.
    ?proj prov:wasAssociatedWith ?pers.
    ?pers  a prov:Person }
```

In this query, constrained to PROV-O terms, we were unable to look for *all* the sensors installed on kirbypost23, *only* the principal investigators of the projects, or *only* those projects where the sensor was used for observing (as opposed to being a photographic subject, for example). Our second query is better targeted, handling these issues by employing the more specific modelling available in both of SSNO and CERIF. It demonstrates enrichment of PROV-O with both sensor-specific and research management-specific concepts of relevance to provenance, through independent alignments of each to PROV-O.

```
SELECT ?proj  ?sys ?device ?pers
    WHERE  {
    ?proj cf:ObservedWith ?sys  .
    ?proj cf:PrincipalInvestigator ?pers .
    {
        {?sys  dul:hasPart  ?device .
        ?device a ssn:SensingDevice .
        ?device ssn:onPlatform  :kirbypost23
        }
        UNION
        {   ?sys a ssn:SensingDevice .
            ?sys ssn:onPlatform  :kirbypost23
        }
    }}
```

These are rather simple queries aimed to compactly demonstrate the value of the mapping. Clearly much more could be retrieved by queries utilising more of the terminology available in PROV-O, SSNO, and CERIF, such as the time period during which the post would have fallen.

6 Conclusion

This paper presented an alignment between the W3C recommendation PROV-O and the current defacto standard for the semantic description of sensors, the SSNO. As well as aligning SSNO concepts and properties to PROV-O, further detail was gained by creating new concepts in the PROV-O hierarchy and linking them to both the SSNO and PROV-O. The alignment links SSNO-based ontologies and observational data to provenance, and is capable of more detailed modelling for sensors than the SSNO alone. In particular, the alignment extends the modelling of how the sensing took place and provides capabilities for detailed modelling of the passage of time in relation to deployments and the changes that take place in installation, maintenance, and upgrade. The extra detail for observation descriptions aligns the SSNO to the O&M view, drawing the two into the same framework and allowing extra interoperability.

Interestingly, not all of the SSNO is aligned to PROV-O. The alignment reflects that provenance describes what has happened, and, hence, PROV-O is not strong on entity to entity relations, meaning that the parts of the SSNO that describe properties of sensors are left unmapped. However, these unmapped properties can still be used to advantage in a provenance context as demonstrated by our second query example. The alignment doesn't align DUL and PROV-O. We felt that the useful alignment was SSNO to PROV-O and that a PROV-O to DUL alignment doesn't provide the same benefit and places restrictions on the meaning of PROV-O that may make the alignment less useful.

Rules provided as part of the alignment further constrain the interpretation of the relationship between the two ontologies. They guide users and implementing tools with using the alignment to define provenance for sensor data. Alternatively, the rules enable SSNO observational data to be automatically enriched with the extra concepts from the alignment and be used in a provenance context.

As sensors continue to become more ubiquitous and the availability of semantic sensor data increases, its use will become even more commonplace than it is today. By documenting data provenance, the alignment described in this paper can play an important role in supporting agents to understand and utilise such data. Given the status of the SSNO and PROV-O as reference models for sensor and provenance descriptions, we believe that this alignment will find service in both communities, and may also act as a guide to others who require to describe the provenance of sensed data.

Acknowledgements The authors would like to thank Keith Jeffery for assistance in interpreting the CERIF data model. David Corsar is supported by the award made by the RCUK Digital Economy programme to the dot.rural Digital Economy Hub (award reference EP/G066051/1).

References

1. Geographic Information – Observations and Measurements. International Standard ISO 19156, International Organization for Standardization (2011)

2. Bisdikian, C., Kaplan, L., Srivastava, M.: On the quality and value of information in sensor networks. ACM Transactions on Sensor Networks 9(4), 48:1–48:26 (Jul 2013)
3. Cheney, J., Chiticariu, L., Tan, W.C.: Provenance in databases: Why, how, and where. Found. Trends databases 1(4), 379–474 (April 2009)
4. Chong, S., Skalka, C., Vaughan, J.: Self-identifying sensor data. In: Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks. pp. 82–93. IPSN '10, ACM, New York, NY, USA (2010)
5. Compton, M., Barnaghi, P., Bermudez, L., Garcia-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowiczj, K., Kelsey, W., Phuoc, D.L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., Taylor, K.: The SSN ontology of the W3C semantic sensor network incubator group. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 17 (2012)
6. Corsar, D., Edwards, P., Baillie, C., Markovic, M., Papangelis, K., Nelson, J.: Short paper: Citizen sensing within a real time passenger information system. In: Proceedings of the 6th International Workshop on Semantic Sensor Networks. vol. 1063, pp. 77–82. CEUR Workshop Proceedings (2013)
7. Cox, S.: Observations and Measurements — Part 1 — Observation schema. OpenGIS Implementation Standard OGC 07-022r1, Open Geospatial Consortium Inc. (December 2007)
8. Cox, S.: Observations and Measurements — Part 2 — Sampling Features. OpenGIS Implementation Standard OGC 07-002r3, Open Geospatial Consortium Inc. (December 2007)
9. Cox, S.: An explicit OWL representation of ISO/OGC Observations and Measurements. In: 6th International workshop on Semantic Sensor Networks. vol. 1063. CEUR-WS (2013)
10. Hensley, Z., Sanyal, J., New, J.: Provenance in sensor data management. Queue 11(12), 50:50–50:63 (December 2013)
11. Janowicz, K., Compton, M.: The Stimulus-Sensor-Observation Ontology Design Pattern and its Integration into the Semantic Sensor Network Ontology. In: 3rd International workshop on Semantic Sensor Networks. vol. 668. CEUR-WS (2010)
12. Jorg, B., Jeffery, K., Dvorak, J., *et. al*: CERIF 1.3 full data model (FDM) introduction and specification. Technical report, euroCRIS (January 2012)
13. Jorg, B., Lappalainen, J., Kastrantas, K.: Modeling the semantics of contextual and content-specific research metadata using ontology languages: issues on combining CERIF and OWL. In: International Conference on Computational Science, ICCS 2012. Elsevier (2012)
14. Lebo, T., Sahoo, S., McGuinness, D., Belhajjame, K., Cheney, J., Corsar, D., Garijo, D., Soiland-Reyes, S., Zednik, S., Zhao, J.: PROV-O: The PROV ontology. W3C Recommendation (2013), available at `http://www.w3.org/TR/prov-o/` (last accessed 23rd April 2014)
15. Ledlie, J., Ng, C., Holland, D., Muniswamy-Reddy, K., Braun, U., Seltze, M.: Provenance-aware sensor data storage. In: Proceedings of the 1st IEEE International Workshop on Networking Meets Databases (NetDB) (April 2005)
16. Lefort, L., Henson, C., Taylor, K., Barnaghi, P., Compton, M., Corcho, O., Castro, R.G., Graybeal, J., Herzog, A., Janowicz, K., Neuhaus, H., Nikolov, A., Page, K.: Semantic Sensor Network XG Final Report. W3C Incubator Report (2011), available at `http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/` (last accessed 23rd April 2014)

17. Liu, Y., Futrelle, J., Myers, J., Rodriguez, A., Kooper, R.: A provenance-aware virtual sensor system using the open provenance model. In: Collaborative Technologies and Systems (CTS), 2010 International Symposium on. pp. 330–339 (May 2010)
18. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E., Van den Bussche, J.: The open provenance model core specification (v1.1). Future Generation Computer Systems 27(6), 743–756 (June 2011)
19. Moreau, L., Ludäscher, D.: Special issue: The first provenance challenge. Concurrency and Computation: Practice and Experience 20(5), 409–418 (April 2008)
20. Moreau, L., Missier, P., Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Klyne, G., Lebo, T., McCusker, J., Miles, S., Myers, J., Sahoo, S., Tilmes, C.: PROV-DM: The PROV Data Model. W3C Recommendation (2013), available at `http://www.w3.org/TR/prov-dm/` (last accessed 23^{rd} April 2014)
21. Park, U., Heidemann, J.: Provenance in sensornet republishing. In: Freire, J., Koop, D., Moreau, L. (eds.) Provenance and Annotation of Data and Processes, Lecture Notes in Computer Science, vol. 5272, pp. 280–292. Springer Berlin Heidelberg (2008)
22. Patni, H., Sahoo, S., Henson, C., Sheth, A.: Provenance aware linked sensor data. In: Proceedings of the 2nd Workshop on Trust and Privacy on the Social and Semantic Web. vol. 5 (May 2010)
23. Sahoo, S., Barga, R., Goldstein, J., Sheth, A., Thirunarayan, K.: Where did you come from...where did you go? An algebra and RDF query engine for provenance. Tech. rep., Kno.e.sis Center, Wright State University (2009)
24. Shebaro, B., Sultana, S., Gopavaram, S., Bertino, E.: Demonstrating a lightweight data provenance for sensor networks. In: Yu, T., Danezis, G., Gligor, V.D. (eds.) Proceedings of the 2012 ACM Conference on Computer and Communications Security. pp. 1022–1024. ACM (2012)
25. Simmhan, Y., Plale, B., Gannon, D.: A survey of data provenance in e-science. SIGMOD Record 34, 31–36 (2005)
26. Stasch, C., Sven Schad, S., Alejandro Llaves, L., Krzysztof Janowicz, K., Broring, A.: Aggregating linked sensor data. In: Taylor, K., Ayyagari, A., De Roure, D. (eds.) Proceedings of the 4th International Workshop on Semantic Sensor Networks. vol. 839, pp. 55–68 (October 2011)
27. Taylor, K., Lamb, D., Griffith, C., Falzon, G., Lefort, L., Gaire, R., Compton, M., Trotter, M., Wark, T.: Farming the web of things. IEEE Intelligent Systems 28(6), 12–19 (November-December 2013)
28. Umuhoza, D., Braun, R.: Trustworthiness assessment of knowledge on the semantic sensor web by provenance integration. In: Proceedings of the 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA). pp. 387–392 (March 2012)
29. W3C OWL Working Group: OWL 2 Web Ontology Language. W3C Recommendation (2009), available at `http://www.w3.org/TR/owl-overview/` (last accessed 23^{rd} April 2014)
30. Welty, C., Fikes, R.: A reusable ontology for fluents in OWL. In: Proceedings of the Fourth International Conference on Formal Ontology in Information Systems (FOIS) (2006)

A Validation Tool for the W3C SSN Ontology Based Sensory Semantic Knowledge

Şefki Kolozali, Tarek Elsaleh, and Payam Barnaghi
e-mail: {s.kolozali, s.elsaleh, p.barnaghi}@surrey.ac.uk

Centre for Communication Systems Research (CCSR)
University of Surrey, Guildford, United Kingdom

Abstract

This paper describes an ontology validation tool that is designed for the W3C Semantic Sensor Networks Ontology (W3C SSN). The tool allows ontologies and linked-data descriptions to be validated against the concepts and properties used in the W3C SSN model. It generates validation reports and collects statistics regarding the most commonly used terms and concepts within the ontologies. An online version of the tool is available at: (http://iot.ee.surrey.ac.uk/SSNValidation). This tool can be used as a checking and validation service for new ontology developments in the IoT domain. It can also be used to give feedback to W3C SSN and other similar ontology developers regarding the most commonly used concepts and properties from the reference ontology and this information can be used to create core ontologies that have higher level interoperability across different systems and various application domains.

1 Introduction

With the advancement of the Internet of Things (IoT) vast amounts of devices will report data based on new applications and services in diverse application domains such as factory optimisation, transport, smart homes and smart cities. According to a report published by Cisco [2] it is predicted that in the next 5-10 years there will be around 50 billion Internet connected devices that will produce 20% of non-video traffic on the Internet. In order to process the IoT data, information management tools that allow effective organisation of data and knowledge representation tools which provide a frame of reference and enable the representation of abstract concepts in a machine-processable way are vital. While IoT devices provide information that are beneficial to diverse application domains, semantic web technologies allow to represent the domain knowledge as a way to handle various forms of heterogeneity and multi-modality by providing semantic models and interoperable data representation forms. Utilisation of semantic technologies for IoT advances interoperability among IoT resources, information models, data providers and consumers. In an effort to agree on a common consensus on a standardisation towards semantic descriptions of sensor

networks an ontology has been developed by the W3C Semantic Sensor Network Incubator group (i.e. W3C SSN Ontology) [1].

Most of the current ontology development methods still require tremendous effort and subjective judgments from the ontology developers to acquire, maintain and validate the ontology. On the one hand, the ability to design and maintain ontologies requires expertise in both the domain of the application and the ontology language used for modelling. However, with their growing utilisation, not only the number of available ontologies increased considerably, but they are also becoming larger and more complex to manage. On the other hand, although there have been numerous work on publishing linked-data on the semantic web and ontology development methodologies in order to transform the art of building ontologies into an engineering activity; ontology and linked-data validation process is another crucial problem since developers need to tackle a wide range of difficulties when modelling and validation ontologies. These difficulties, such as the appearance of anomalies in ontologies or the technical quality of an ontology against a frame reference plays a key role in the ontology engineering projects.

The purpose of this study is to describe and examine the validation issues of sensory information in the IoT domain, and analyse various terminologies in order to provide assistance in the ontology development process. Thus, we propose the W3C SSN ontology validation service, which is based on Eyeball validator to check the RDF descriptions, to enable a user to validate an ontology or linked data on various common problems including use of undefined properties and classes, poorly formed namespaces, problematic prefixes, literal syntax validation and other optional heuristics. Moreover, enabling validation of Linked IoT data descriptions against W3C SSN ontology, we allow users to detect domain specific semantic and factual mistakes that may need an overview of a domain expert. This can help an effective integration of domain specific ontologies into linked-data models. We also collect and present information regarding the popularity of terms that are used by ontologies and the IoT data submitted for validation. This work is developed in the context of the CityPulse project[3][1]. The remainder of this paper is organised as follows. Section 2 describes the SSN validation web tool. Section 3 details the evaluations in which we investigate the most popular terms and modules that have been used within the W3C SSN ontology by examining various SSN related available ontologies. Finally, in the section 4, we note on further challenges of the semantic modelling for the IoT data and outline our future work.

2 The SSN Validation Tool

Fig. 1 presents the architecture of the SSN Validation Tool. The W3C SSN validation web application integrates the ontology and data validation functionalities in a web-based client-server architecture. The client runs in most popular

[1] This work is supported by the EU FP7 CityPulse Project under grant No.603095. http://www.ict-citypulse.eu

web browsers, and provides an easy to use interface. It allows the user to interact with the application and perform the following actions: *i*) enter an RDF document into text box or upload via a browse button to be validated against a reference ontology (i.e. in this case the W3C SSN ontology) *ii*) retrieve a list of evaluation results, *iii*) select and see namespaces of a term from the tag clouds as one would from a search engine and also visualise the most common terms and concepts used in the ontologies.

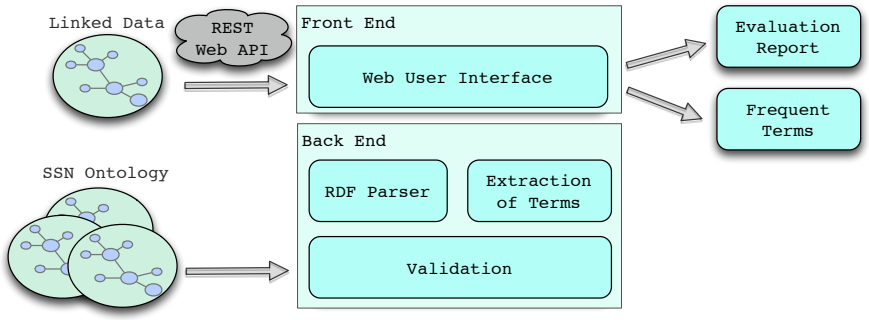


Fig. 1: The architecture of the SSN Validator web application

2.1 Front-End

The validator application is developed using Java EE, HTML, JSP technologies. It takes an RDF input or an ontology in order to produce a set of evaluation results. The web user interface consists of a single view where the user enters the RDF data into text box or uploads via a browse button. In response to user interaction, the server performs the core functionalities as shown in Fig 1. Concerning client-server communication, the validator follows the Representational State Transfer (REST) style web application design. In this architecture web pages form a virtual state machine, allowing a user to progress through the application by entering or uploading an RDF document which results in a transition to the next state of the application by transferring a representation of that state to the user.

2.2 Back-End

There are three main functionalities in the main system, namely RDF Parser, Extraction of Terms, and Validation. Initially, the RDF document describing the ontology or RDF document is parsed using the Jena API to obtain RDF triples as an input to the validation system. The server side of the SSN Validation web

application builds on the Eyeball validator, which is a Java library for validating RDF documents. This is extended with modules to domain specific analysis for the W3C SSN ontology. It scans for errors from those available in the Eyeball list regarding RDF, Turtle and N3 syntax and some modelling suggestions are also generated.

The validation results are displayed by means of the web user interface showing a list of errors, and explanations regarding the ontology elements affected. The application also reports the recurrence of terms that are not present within the W3C SSN ontology. We have developed a server-side JavaScript code which interacts with the embedded Tag Cloud to display extracted terms that are not present in the W3C SSN ontology. The terms requested when the user starts the validation process, and returned using JavaScript Object Notation with evaluation results, which is presented at the same time with extracted terms as tag cloud. The term recurrence tags are displayed using an adapted version of the WP-Cumulus Flash-based Tag Cloud plug-in. This plug-in utilises XML data generated on the server side from the extracted terms. The light-weight client uses a combination of standard web technologies (HTML, CSS, JavaScript) and uses a Java library to dynamically load content from an object oriented database (i.e. DB4o).

3 The Ontology Validations

We collected a set of available ontologies and semantic description models that report using and/or extending the W3C SSN ontologies. The ontology dataset includes the Smart Product Ontology[2], the SPITFIRE project ontology[3], The IoT.est project service model[4], The SemSorGrid4Env project ontology[5], The OntoSensor ontology[6], The WSML Event Observation ontology[7], The WSML Environment Observation Ontology[8].

We evaluated these ontologies using our validation tool to find out the noise, inconsistency and syntax errors along with the similarity between the W3C SSN concepts and the terms and concepts used in these ontologies. It can be difficult for an ontology engineer to identify some errors and unexpected incorrect inferences in RDF. In the RDF data model, terms are typically named by Web URIs, which may be dereferenced to access more information such as vocabulary definitions about their meaning. However, while the principal notion behind the Semantic Web is to experience a machine-oriented world of Linked Data,

[2] http://www.w3.org/2005/Incubator/ssn/wiki/SSN_Smart_product
[3] http://spitfire-project.eu/ontology.owl
[4] http://personal.ee.surrey.ac.uk/Personal/P.Barnaghi/ontology/OWL-IoT-S.owl
[5] http://www.semsorgrid4env.eu/ontologies/CoastalDefences.owl
[6] https://www.memphis.edu/eece/cas/onto_sensor/OntoSensor.txt
[7] https://code.google.com/p/wsmls/source/browse/trunk/global/Observations/0.2/Observation.n3?spec=svn70&r=70
[8] https://code.google.com/p/wsmls/source/browse/trunk/global/Event-observation/0.2/EventObservation.n3?spec=svn207&r=207

ontology engineers should be very cautious to prevent broken links as well as make URIs dereferencable in order to empower automatic data access for Semantic Web applications. In accordance with the use of HTTP URIs, we found in our validations that in some instances (i.e. WSML event and WSML environment) different URIs were utilised rather than primary resources. As a result, it redirects the application user to their local directory instead of original locations such as SSN Ontology. Some of other errors were identified in IoT.est model and SemSorGrid4Env, in which multiple prefixes were defined (i.e. `owl` and `CoastalDefences`, respectively), in addition to utilisation of upper-case in namespaces of IoT.est model (i.e. `http://www.loa-cnr.it/ontologies/DUL.owl#`). It is interesting to see that while the latter is not actually wrong, it is accepted as unconventional and pointless for eyeball tool.

In parallel, sometimes properties or classes are used without any formal definition. In SPITFIRE, for instance, it has been defined that `:savedEnergyOf rdfs:domain :SavedEnergy`, even though `:SavedEnergy` is not defined as a class. Nevertheless, although such practice is not prohibited, such ad-hoc undefined classes and properties make automatic integration of data less efficient and prevent the possibility of making inferences through reasoning. An additional error that has been found for SPITFIRE via our validation tool is a syntax error where `ssn:subPropertyOf` was used instead of `rdfs:subPropertyOf`. Finally, we discovered in OntoSensor that there was clearly a misuse of functional property syntax along with a data property. It needs to be updated in line with OWL-2 guidelines using `FunctionalDataProperty` that describes properties for each individual allowing for at most one distinct literal.

Table 1: Summary of ontology evaluations against the W3C SSN ontology. Similar terms: s-terms; Dissimilar terms: d-terms; Similar properties: s-prop; Dissimilar properties: d-prop; Similar concepts: s-concept; Dissimilar Concepts: d-concept

	s-terms	d-terms	s-prop	d-prop	s-concept	d-concept
Smart Product	12	25	11	5	10	11
SPITFIRE	2	94	0	67	3	26
IoT.est model	0	12	0	10	0	2
SemSorGrid4Env	2	31	1	3	2	27
OntoSensor	0	331	0	226	0	105
WSML event	0	7	0	0	0	7
WSML environment	0	7	0	0	0	7

Table 1 summarises the results of similarity of terms and shows statistics using the W3C SSN ontology concepts in these ontologies. We found that the most frequently used SSN terms are as follows: `Property`, `Device`, `Observation`, `FeatureOfInterest`, and `ObservationValue`. Based on the validation results, we also created a Tag Cloud that shows the most common concepts that are used in the validated ontologies. We also checked linked ontologies and other common description models that can be used in the form of linked-data. Considering the

most common concepts and properties that are used from the W3C SSN ontology can help to create an optimum core ontology in which the main concepts and properties are used in several other related ontologies. This can also give an indication of which parts of the SSN ontology are used more than others. The latter can provide feedback to the ontology developers to help them focus on the most commonly used features and create automated tools and software that can enhance and increase interoperability across different applications and systems in the IoT domain.

4 Conclusions

This paper describes a validation tool that is mainly designed for the W3C SSN ontology. However, it can be also used with other base line ontologies to validate linked-data descriptions and ontologies against reference ontologies. As the number of semantic models and description frameworks in the IoT domain increases, interoperability between the various models becomes an issue. The W3C SSN ontology is designed to describe sensor networks and device related features. This ontology has been used and extended in several projects and applications. We have created an online tool to validate semantic models and linked-data descriptions against the W3C SSN ontology. We used the tool to validate a set of ontologies that are available online in which the W3C SSN ontology was used as a base ontology. We created a Tag Cloud and presented the most common terms and concepts that are used from the W3C SSN ontology and provided statistics regarding the number of concepts and properties that are adapted from the SSN model in each of the ontologies.

The validation service can be used for checking and evaluating the new ontologies against a base line ontology; i.e. the W3C SSN ontology. It can be also used to collect information and statistics about the use of the W3C SSN ontology and provide feedback to the ontology developers. Future work will focus on automated matching and ontology alignment to improve interoperability between different ontologies that are developed in the IoT domain.

References

1. Michael Compton, Payam Barnaghi, Luis Bermudez, Raul García-Castro, Oscar Corcho, Simon Cox, John Graybeal, Manfred Hauswirth, Cory Henson, Arthur Herzog, et al. The ssn ontology of the w3c semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:25–32, 2012.
2. Dave Evans. The internet of things: How the next evolution of the internet is changing everything. Technical report, Cisco Internet Business Solutions Group (IBSG), April 2011.
3. R. Tonjes, M. I. Ali, P. Barnaghi, S. Ganea, F. Ganz, M. Haushwirth, B. Kjargaard, D. Kumper, A. Mileo, S. Nechifor, A. Sheth, V. Tsiatsis, and L. Vestergaard. Real time iot stream processing and large-scale data analytics for smart city applications. *European Conference on Networks and Communications*, 2014.

Weather Station Data Publication at Irstea: an Implementation Report

Catherine Roussey[1], Stephan Bernard[1], Géraldine André[1], Oscar Corcho[2], Gil De Sousa[1], Daniel Boffety[1], and Jean-Pierre Chanet[1]

[1] Irstea, UR TSCF Technologies et systèmes d'information pour les agrosystèmes, 9 avenue Blaise Pascal - CS 20085 Aubiére, F-63178, France
{firstname.lastname}@irstea.fr
[2] Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
ocorcho@fi.upm.es

Abstract. We describe how we are publishing RDF data about one of the meteorological stations that Irstea owns in its experimental farm at Montoldre. Our objective is to revisit previous work done by other researchers on the publication of meteorological Linked Data, and provide some recommendations and best practices based on our experience.

Keywords: SSN, meteorological data

1 Introduction

Several works in the state of the art address the publication of meteorological data as Linked Data, using the Semantic Sensor Network (SSN) ontology [8] proposed by the W3C Incubator Group on Semantic Sensor Networks. The work presented in this paper aims to confirm whether the steps taken by these previous works on publishing meteorological data can be easily reused for a similar purpose and to detect any potential difficulties in the usage of the SSN Ontology, rather than presenting new innovations on the application of the SSN Ontology in this domain.

We have chosen to publish data from the Vantage Pro 2 weather station in use at our experimental farm located in Montoldre (France)[3]. We have followed the usual steps in Linked Data publication, as discussed in [1], paying special attention to the reuse of existing general and domain ontologies applicable to this type of data publication.

This paper is organized as follows: first, we describe our weather station and its data; next we study some works on weather data publication based on SSN. Then we briefly present the network of ontologies that we have reused. In Section 5, we describe how we have populated the selected ontology network, and Section 6 describes the workflow followed in this process. Finally we conclude by presenting an analysis of our work and perspectives.

[3] http://www.irstea.fr/la-recherche/themes-de-recherche/motive/station-de-montoldre

2 Montoldre's Weather Station Description and Data Sources

Irstea has a research and experimentation site located at Montoldre, where different types of experiments are run. This site has its own weather station, a Vantage Pro 2[4] from Davis Instruments[5]. The station has the following components: a clock, two temperature sensors (indoor and outdoor), an atmospheric pressure sensor, two air humidity sensors (indoor and outdoor), a wind vane, an anemometer, a rain gauge to measure water precipitation and speed, and a solar radiation sensor.

The external sensors communicate wirelessly with a console located inside a building, which (in addition to the sensors it contains) allows calculating other variables (wind chill, dew point, etc.), as well as weather forecasts. By connecting the console to a computer it is possible to store the values measured and put them online on a Web server[6]. The storage of measures is automatically done in the station according to the user parameters (intervals of time, units, etc.). These data can be extracted to generate a CSV (comma-separated values) file containing the following information:

Date	Time	Temp Out	Hi Temp	Low Temp	Out Hum	Wind Speed	Wind Dir
01/01/13	0:30	6.2	6.7	6.0	73	1.6	WSW
01/01/13	1:00	7.1	7.1	6.1	68	3.2	WSW
01/01/13	1:30	7.5	7.5	7.0	67	3.2	WSW
01/01/13	2:00	7.8	7.8	7.5	67	4.8	WSW
01/01/13	2:30	7.7	7.8	7.7	70	4.8	WSW
01/01/13	3:00	7.2	7.7	7.2	75	3.2	SW

For the purpose of data publication, we have generated CSV files for the period between 2010 and 2013 with the following measures: outdoor temperature, external atmospheric pressure, relative humidity of outside air, wind direction, wind speed, precipitation quantity, water precipitation rate, and solar radiation.

3 State of the Art

The SSN ontology [8] can be used as the basis for the publication of weather station data. This ontology must be linked with other ontologies to form an ontology network, including the following ones:

- Ontologies to describe the different type of sensors.
- Ontologies to describe weather phenomena and their measurable properties.
- Ontologies to describe units of measurement.

[4] http://www.davis-meteo.com/Vantage-Pro2.php
[5] http://davisnet.com/
[6] http://meteo.clermont.cemagref.fr/ - service accessible only inside Irstea

– Ontologies to describe geographical places and their location.
– Ontologies to describe temporal entities.

Table 1 lists some of the datasets that use SSN for publishing meteorological data as Linked Data. The first column indicates the name of the dataset. The column "Environmental ontology" presents the ontology used to describe the Feature Of Interests and their associated properties. The column "Spatial ontology" indicates the ontology used to describe the localisation information. The column "time ontology" indicates the ontology used to describe time information, if any. The last column shows more ontologies used in the dataset.

dataset name	environmental ontology	spatial ontology	time ontology	other ontologies
AEMET	AEMET	WGS84 Geobuddies	W3C Time	
Swiss Experiment	SWEET	WGS84		QUDT
ACORN-SAT		WGS84	UK Intervals	DUL Data Cube
SMEAR	SWEET	Geoname WGS84	DUL	Data Cube Situation Theory

Table 1. Projects where SSN has been used to publish meteorological data

AEMET (Agencia Estatal de Meteorología) is the Spanish public agency in charge of collecting and publishing weather data. The workflow used to publish this dataset as Linked Data is described in [9]. It is important to note here that this work was done in parallel to the development of the SSN Ontology, and for this reason some of the design decisions taken for the publication of these data sources are not totally compliant with the current SSN Ontology. This dataset uses the ontologies: AEMET[7], WGS84[8], Geobuddies[9] and W3C Time[10].

The Swiss Experiment Linked Data publication effort reported in [10] proposes the use of the SSN ontology to combine and publish several meteorological data streams provided by heterogeneous sensor networks. Thus this work combines several ontologies to build the common schema that will be used to query sensor data and metadata. SWEET[11], WGS84 and QUDT[12] are reused in this project.

The Australian Bureau of Meteorology published an homogenised daily temperature dataset called ACORN-SAT. In [12], the authors describe how this

[7] http://aemet.linkeddata.es/models_en.html
[8] http://www.w3.org/2003/01/geo/
[9] http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/ontologies/83-geobuddies-ontologies
[10] http://www.w3.org/TR/owl-time
[11] http://sweet.jpl.nasa.gov/
[12] http://www.qudt.org/

dataset was published on the Linked Data using two main ontologies: SSN and RDF Data Cube[13]. UK Intervals[14] and DUL[15] are also used.

The Finnish Station for Measuring Ecosystem Atmosphere Relations (SMEAR) is a large scale sensor network which measures environmental phenomena like weather or atmospheric gases. The works presented in [13] propose a software framework able to interpret sensor data at different level of details. The global architecture is composed of hierarchical layers : measurement, observation, derivation and situation. Each layer reads the data from previous one and increases data complexity in order to enhance its interpretation. The observation layer is based on the SSN ontology.

To conclude, table 1 shows that all meteorological datasets use different networks of ontologies. Thus SSN is generic and does not prevent to build heterogeneous schema for meteorological data publication.

4 An Ontology Network for Weather Station Data Publication

In this section we describe the ontologies that we have reused for the publication of our weather station data. We describe briefly these ontologies and the main parts that we have used from them, if any.

4.1 The W3C Semantic Sensor Network (SSN) Ontology

The *"Semantic Sensor Network"* (SSN) ontology [8] is a generic ontology created by the W3C Semantic Sensor Network Incubator Group. This ontology contains different modules. Given our interest on the publication of sensor data and on describing the sensors used to produce such data, we have focused on the classes and properties defined in the following modules: Skeleton, Data, Platform Site, Device and System. More specifically, the classes that we have reused from this ontology are:

- $ssn : Observation$ to describe the measurement context,
- $ssn : FeatureOfInterest$ to specify the observed phenomena,
- $ssn : SensingDevice$ to describe the sensors,
- $ssn : Platform$ to describe where the sensors are installed,
- $ssn : System$ to describe a system composed of several sensors such as, for example, our weather station.

We have also used the main properties associated to these classes: $ssn : observedProperty$, $ssn : observedBy$, $ssn : hasSubsystem$, $ssn : featureOfInterest$, etc.

[13] `http://purl.org/linked-data/cube`
[14] `http://reference.data.gov.uk/def/intervals`
[15] `http://www.loa-cnr.it/ontologies.DOL.owl`

4.2 The AWS Ontology for Meteorological Sensors

The *"Ontology for Meteorological Sensors"* [6] (AWS) extends the SSN ontology by extending its class $ssn : SensingDevice$. It is focused on the description of different models of sensors that can be used to measure weather phenomena, and hence is also of interest for our purposes.

More specifically, we have mostly reused the following classes:

- $aws : AtmosphericPressureSensor$ for the atmospheric pressure sensor,
- $aws : CapacitiveThinFilmPolymer$ specialisation of $aws : HumiditySensor$ for the hygrometer,
- $aws : Pyranometer$ specialisation of $aws : RadiationSensor$ for the solar radiation sensor,
- $aws : Thermistor$ specialisation of $aws : TemperatureSensor$ for the thermometer,
- $aws : TippingBucketRainGaugeTbrgWithoutCorrection$ specialisation of $PrecipitationSensor$ for the pluviometer; this sensor is able to produce two separate measurements: the quantity of rain and the speed of the precipitation,
- $aws : WindVane$ specialisation of $aws : WindSensor$ for the weather vane,
- $aws : CupAnemometer$ specialisation of $aws : WindSensor$ for the anemometer,

Furthermore, some of these classes contain restrictions indicating which type of $ssn : Property$ they are able to measure. For example, the class $aws : Pyranometer$ is defined by $Pyranometer \sqsubseteq \exists observes.EnergyFlux$. This has been also useful for our purpose. Note that sometimes the documentation of the weather station is not complete enough in order to select the appropriate sensor model in AWS hierarchy. This is the case of the atmospheric pressure sensor. We were not able to select one of those sensors. Thus we do not try to specialise this sensor. We have noticed that AWS proposes lots of sensor models. In our case AWS provides all the sensor descriptions we need for our purpose. Note that AWS does not import any ontology but it defines and reuses several prefixes like *dim* of the QU ontology.

4.3 Climate and Forecast (CF) features Ontology

The *"Climate and Forecast features"* ontology [2] (aka $cf - feature$) is a translation of the "Climate and Forecast (CF) standard names vocabulary"[16] maintained by the "Program for Climate Model Diagnosis and Intercomparison"[17]. This ontology was used to produce one of the use cases of the W3C SSN ontology.

The ontology $cf - feature$ proposes elements to describe climate measurements and weather phenomena. It consists of two modules. The module "cf-feature" is used to define environmental observed phenomena (rain, wind, etc.). It thus contains classes that specialise the $ssn : FeatureOfInterest$ class. We have used the following individuals and classes from this ontology:

[16] `http://cf-pcmdi.llnl.gov/documents/cf-standard-names/`
[17] `http://cf-pcmdi.llnl.gov/`

- $cf - feature : air$ instance of $cf - feature : Medium$,
- $cf - feature : ground\_level\_soil$ instance of $cf - feature : SurfaceMedium$,
- $cf - feature : rainfall$ instance of $cf - feature : Precipitation$,
- $cf - feature : wind$ instance of $cf - feature : Wind$.

The module "dim" is used to define measurable properties (speed, volume, etc.). It contains classes that should specialise the $ssn : Property$ class[18]. Note that this module uses the prefix dim of the QU ontology. All the individuals of this module are defined with the prefix $cf - property$. We have reused the following individuals:

- $cf - property : air\_temperature$ instance of $dim : Temperature$,
- $cf - property : air\_pressure$ instance of $dim : StressOrPressure$,
- $cf - property : relative\_humidity$ instance of $dim : Dimensionless$,
- $cf - property : wind\_from\_direction$ instance of $dim : Angle$,
- $cf - property : wind\_speed$ instance of $dim : VelocityOrSpeed$,
- $cf - property : rainfall\_amount$ instance of $dim : SurfaceDensity$,
- $cf - property : rainfall\_rate$ instance of $dim : VelocityOrSpeed$,
- $cf - property : downward\_heat\_flux\_at\_ground\_level_in_soil$ instance of $dim : EnergyFlux$.

Some of these individuals are linked to individuals of the module "cf-feature". For example, $cf - property : air\_pressure$ is linked to $cf - feature : air$ by the property $ssn : isPropertyOf$. However, this is not the case for all individuals (for example, $cf - property : relative\_humidity$ is not linked to any instance of $ssn : FeatureOfInterest$). Hence we had to provide some of these links, as shown in the Table 2.

In general, this ontology still needs to be better documented. We have difficulty to make the distinction between $cf - feature$ ontology and $cf - property$ one, these two ontologies having very similar schema. Thus for clarity purpose maybe only one ontology should be defined. Moreover, sometimes it is difficult to choose between two individuals. This is, for example, the case, for $cf - feature : air$, which is an instance of $cf - feature : Medium$, and $cf - feature : atmosphere\_air$, which is an instance of $cf - feature : LayerMedium$. As this ontology also imports many other ontologies, it is sometimes difficult to know the origin of some of the elements that it defines, in order to make decisions about their usage or reuse.

4.4 The Library for Quantity Kinds and Units (QU)

The *"Library for Quantity Kinds and Units"* (QU) ontology [4] has been also created by a W3C working group. The $cf - feature$ ontology, discussed in the previous section, imports directly the QU ontology. Indeed, the $cf - feature$ ontology reuses the modules "dim" and "unit" of the QU ontology. The AWS

[18] All these classes are defined as subClassOf $qu : QuantityKind$. They are not defined directly has subClassOf $ssn : Property$

ontology does not import QU but reuses its prefixes like *dim*. The module "dim" of the QU ontology defines classes that are useful to categorise physical quantities, such as $dim : Angle$, $dim : Distance$, $dim : Dimensionless$, etc. The module "unit" defines classes that are useful to categorise measurement units, and also provides instances of those classes, so as to identify units such as $unit : hectopascal$, $unit : percent$, etc. Therefore, we have reused this ontology[19] for the representation of our units of measurement, as we will discuss later.

4.5 The ISA Location Core Vocabulary (LOCN) and GeoSPARQL

Currently, several ontologies exist for the publication of spatial data. Each of which has a different origin and different purposes. This makes it sometimes difficult to determine which are the best options to follow when aiming at representing geospatial information, such as the one associated to a weather station.

The WGS84 vocabulary is the oldest and most commonly used vocabulary to indicate the spatial coordinates of any geographical feature. All the previous work on publishing meteorological dataset use it. We decide not to use it because all these datasets do not contain the location property that link a spatial thing to the point which is its geometrical representation. We think that a clear distinction should be made between the spatial object and one of its possible geometrical representations.

We decide to use the *"GeoSPARQL"* vocabulary [5]. GeoSPARQL is the result of a standardization process at the Open Geospatial Consortium (OGC), which was initially focused on querying geographical data. It has also proposed the model to use for describing geometries of spatial objects (through the object property *hasGeometry*, and the use of GML or WKT strings). GeoSPARQL extends the WGS84 vocabulary and proposes different types of geometries like: point, polygon, multipolygon, etc. It also allows defining topological relationships between geometric elements.

The *"ISA Core Location"* vocabulary [3] (LOCN) was released in November 2013, and has recently been given a W3C-owned namespace, although it was initially generated outside the consortium. This lightweight ontology is focused on the description of places and their address, providing a set of three classes and several properties for their description. Notably, aspects related to the geometry description of places are still in an *"unstable"* state, and hence they may change in the future. This is the ontology that we have used for specifying the address of the experimental farm.

4.6 The W3C Time Ontology

The W3C Time ontology [7][11] enables the description of time instants (instances of the class $time : Instant$) and intervals (instances of the class $time : Interval$). Hence it may be useful when we need to describe the timestamps or the time intervals associated to the measurements made by the weather station.

[19] prefixed qu-rec20 `http://purl.oclc.org/NET/ssnx/qu/qu-rec20`

We have used the classes $time : Interval$ and $time : Instant$, and the associated properties $time : month$, $time : hour$, etc.

5 Design and Population of the Ontology Network for Weather Station Data

Based on the ontology network described in the previous section, we are now able to create a dataset containing all the individuals describing measurements of our weather station. Now we explain the decisions taken in order to create resource URIs (Section 5.1) and we provide examples of how we publish different types of data according to our ontology network (Section 5.2).

5.1 Resource URIs for our Weather Station Data

URIs have been designed with several principles in mind, such as simplicity, stability and manageability. We have followed common guidelines for their effective uses, following in many cases the recommendations already applied in [9]. This section presents the main URI design decisions and conventions used, and Table 3 provides a summary of the main types of URIs that we generate.

The base URI is `http://ontology.irstea.fr/weather/`, prefixed as *irstea*. Hence all individuals follow the URI scheme `http://ontology.irstea.fr/weather/resource`. For example, the URI to identify the experimental research site of Montoldre is: `http://ontology.irstea.fr/weather/resource/location#irsteaClermontMontoldre`.

Specially relevant is the template used for generating identifiers for the observations (that is, instances of the class $ssn : Observation$). In this case, we had initially considered moving from the URI template that had been used for the AEMET Linked Data generation (where cool but rather long URIs were generated as a consequence of including in the observation the identifiers of the timestamp, sensor and property that was measured) into more simple URIs generated as an MD5 hash code from the string proposed in Table 3. For instance, the MD5 hash code for that observation would be something like *eea6c7338102cb8866c8ad563bb85faf*. After discussing with our domain experts, they did not find it so troublesome to have long URIs with a descriptive identifier, since they considered that this is a rather normal way to name files in many file systems for many scientific domains. Hence we kept them as cool URIs.

5.2 Excerpts of our Ontology-based Weather Station Data

In the following subsections we provide examples of how we generate some of our RDF data according to the selected ontologies. We think that these excerpts of the global ontology instances will be useful for others trying to generate and publish RDF data from a similar domain. It is also a good example of the use of SSN with other ontologies.

Describing the Weather Station. In this section we provide a general overview of how we describe the main context of where our weather station is located. Besides the SSN Ontology (our weather station is an instance of the $ssn:System$ and $ssn:Platform$ classes) we use two others vocabulary: the GeoSPARQL vocabulary in order to relate the weather station to its geometry, and the ISA Core Location Vocabulary (LOCN) for the description of the address where the Montoldre experimental farm is located. Note that the $geosparql:Geometry$ instance uses the $geosparql:asWKT$ property, with a WKT string to specify the corresponding geographical point. Note that the LOCN vocabulary propose to use the $dcterms:Location$ class to identify spatial objects.

To combine the LOCN vocabulary with the GeoSPARQL one, we add an instantiation link from the $dcterm:Location$ instance to the $geosparql:SpatialObject$ class.

Figure 1 provides a graphical summary of our model and individuals.

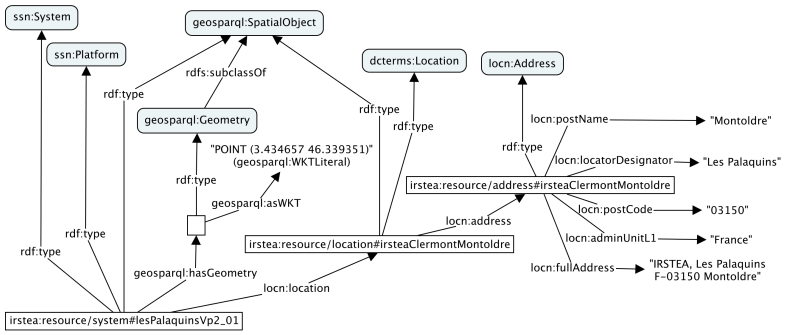


Fig. 1. Description of our weather station at Montoldre

Describing the Sensors deployed in the Weather Station. We have also focused on describing each of the sensors that are assembled in our weather station. Figure 2 provides a graphical overview of one of these sensors, more specifically the anemometer.

This anemometer is identified by a URI that finishes with $WindGauge\_01$. It is an instance of the class $aws:CupAnemometer$, which is a subclass of $aws:WindSensor$, in order to specify clearly the type of sensor that the anemometer represents. This anemometer is a subsystem of the weather station, expressed with the property $ssn:hasSubsystem$ between the weather station and the sensor. Finally, we specify that the sensor observes the individual $cf-property:wind\_speed$.

An important aspect to note here is to be able to connect the individuals of $cf-feature$ ontology to the class $ssn:Property$. The definition of $ssn:Sensor$ states that a sensor observes only instance of $ssn:Property$. Thus a reasoner

will deduce automatically that the individual $cf - property : wind\_speed$ is an instance of $ssn : Property$. Even if the class $qu : QuantityKind$ is not defined as a subclass of $ssn : Property$, we can deduce that, in the case of our description of sensors, any instance of $qu : QuantityKind$ will also be an instance of $ssn : Property$. This inference is represented in figure 2 by a dash arrow.

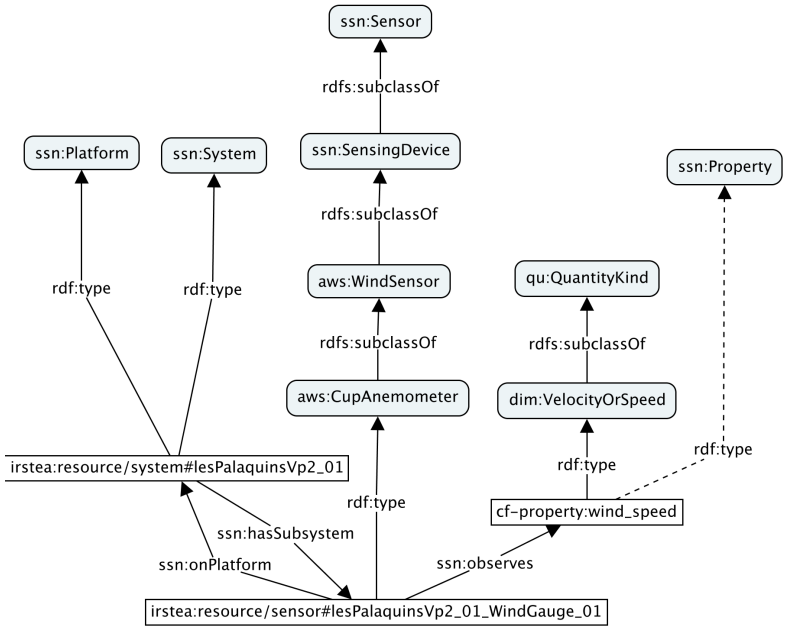


Fig. 2. Description of the wind gauge installed in our weather station

Describing the Observations. Observations allow describing the context of a measurement done by a sensor, and their description lies at the core of the SSN Skeleton module, and it is one of the most well documented patterns to follow. Figure 3 represents an observation done by the anemometer, described previously, on the wind speed at a given point in time. We can see here that we use the properties $ssn : observedProperty$, $ssn : featureOfInterest$, $ssn : observedBy$ and $ssn : observationResult$ to relate our specific observation with its corresponding observed property, feature or phenomenon, sensor used to obtain the measurement, and result of the measurement, respectively. As for the result of the observation, it is important to note that this is a pattern that is not so well documented and hence there are multiple ways to represent the observation values, together with their corresponding units. For example, the QU ontology

proposes the properties $qu : numericalValue$ and $qu : unit$. As shown in Figure 3 we prefer to use the properties already contained in ssn:

- the property $ssn : hasValue$ to relate the result with the corresponding $ssn : ObservationValue$,
- the property $DUL : hasDataValue$ to express the actual value,
- the property $DUL : isClassifiedBy$ to express the unit of the measurement.

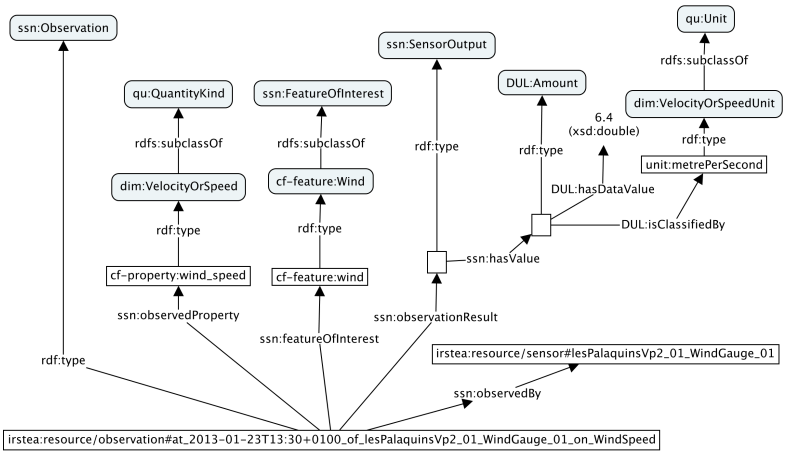


Fig. 3. Description of an observation of wind speed

Describing the Time Instants or Intervals associated to the Observations. This is also a pattern that is not well documented in the existing SSN Ontology documentation, and hence many options are available for the generation of the corresponding data. An observation can be related to a time instant or describe the result of an aggregation of values (average, sum, maximum, minimum, etc.) over a time interval. The relationship of the observation with the corresponding time instant or time interval is done with the property $ssn : observationResultTime$, and the terms that are needed to specify time instants and intervals are provided in the W3C Time Ontology.

Figure 4 presents an example where the observation is related to a time instant (for instance, in the case of a temperature measurement). In this case, the range of the property $ssn : observationResultTime$ is an instance of the class $time : Instant$ which is connected to the corresponding $xsd : dateTime$ value according to the ISO 8601 format via the property $time : inXsdDateTime$. Besides, we provide in our examples additional properties defined by the W3C Time Ontology, such as $time : year$, $time : month$, $time : day$, $time : hour$, and $time : minute$, so as to allow creating GROUP BY queries more easily, which

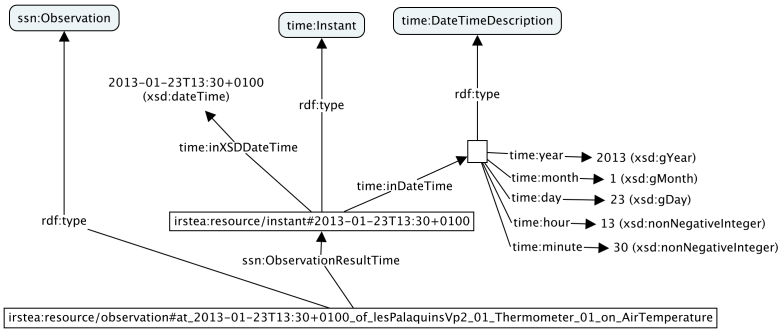


Fig. 4. Description of an observation related to a time instant

can be used for aggregation purposes by data consumers when accessing our SPARQL endpoint.

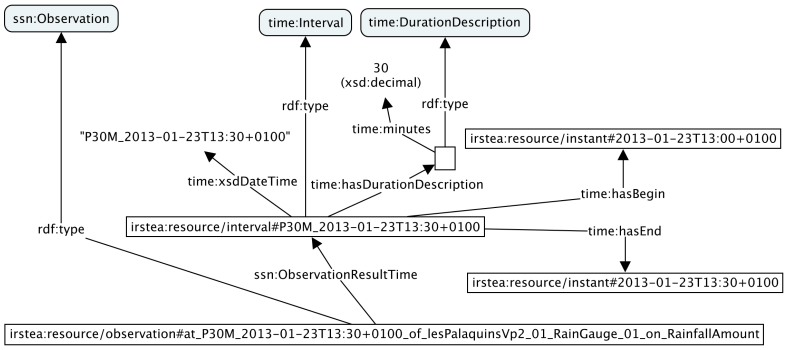


Fig. 5. Description of an observation related to a time interval

Figure 5 presents an example where the observation is related to a time interval (for instance, in the case of rainfall amount). In this case, we follow a similar pattern, but the value of the property $ssn : observationResultTime$ is instead an instance of the class $time : Interval$, and we also use the properties $time : hasDurationDescription$ and $time : hasEnd$ to specify the duration of the interval and its end, besides the beginning of the interval.

Table 4 contains a summary of the type of timestamp that we have associated to each type of measurement that our weather station sensors perform.

6 Implementation of the Data Transformation Process

Now we describe briefly the process followed for the generation of the RDF data according to the design principles described in the previous section. All the data that we have generated are available at our SPARQL endpoint[20].

As explained in the Introduction, data measured by the weather station sensors are stored in CSV files. These measurements are performed every half or quarter an hour. Figure 6 shows the main data processing flow that we apply.

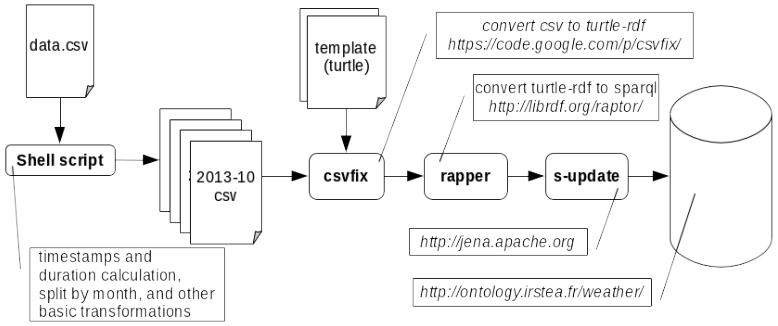


Fig. 6. Data Processing Flow for the Generation of RDF Data from the Weather Station Measurements

Pre-processing and date-related operations. A first pre-processing step is done using shell functions, which allows easy and quick conversions and substitutions, such as deleting line headers, detection and removal of ill-formed lines, and the substitution of initial tab characters by a semicolon, better accepted by the next tools in the pipeline (e.g., the csvfix tool). Moreover, we used standard Unix tools, for the following purposes:

- to convert dates as they were stored in the CSV file into the ISO 8601 format,
- to calculate the time interval between two timestamps,
- to convert the wind direction in degree.

The last pre-processing step is to group data by month, as a pragmatic way of processing batches of data and allowing an easier load into our Fuseki server.

CSV to RDF Transformations. The conversion of CSV data to RDF Turtle is made using the *csvfix*[21] toolbox for CSV file, which allows converting CSV data to other formats using template files.

[20] `http://ontology.irstea.fr/weather/query/`
[21] `https://code.google.com/p/csvfix/`

We have created some RDF Turtle format template files which are used in this process[22]. Finally, the data are sent to the server using *s-update*, which is part of SOH (SPARQL over HTTP) from the Jena framework[23].

7 Conclusions, Perspectives and Recommendations

As discussed in the Introduction, our intention with this work lied mainly at providing an implementation report about the usage of a set of ontologies that have been so far associated to the generation and publication of Linked Data from weather stations. This work tries to reflect which parts of the existing documented examples need to be refined or extended, and which parts are already well described and reproducible for a weather station.

We hope that our examples and code can provide further information to people that will be involved in a similar processing pipelines. Besides that, we have the following recommendations for the following versions of this set of ontologies:

- In general, we think that we still need a better documentation for many of these ontologies, in order to facilitate their use. This covers both documenting some of the classes and properties that are defined in these ontologies and providing more examples of usage. This does not say that the current documentation is bad at all. However, the documentation is still sometimes difficult to understand by people who are not specialists in ontologies. It would be interesting to associate it with concrete examples showing good practices in publication of data collected by sensors, such as what we try to do in this paper, or what is available in many other papers as well as in the W3C SSN Incubator Group implementation report. The lack of documentation is also a limitation of the $cf-feature$ ontology, as discussed in this paper, what makes it sometimes difficult to take decisions about which instances of properties to use for a specific type of measurement.
- There is still too much heterogeneity in ontologies that can be used to express geospatial information (W3C WGS84, GeoSPARQL, the ISA Core Location Vocabulary, etc.). This forces us to spend a lot of time deciding which ontologies to use for describing each type of geospatial information (locations, spatial points, polygons, etc.), and in some cases many alternatives exist, what may make some applications and SPARQL queries to work for some SPARQL endpoints and fail in others.
- The $cf-feature$ and $cf-property$ ontologies should be merged in one ontology in order to clarify their usage.
- The pattern to use for the specification of time instants and intervals associated to observations has to be unified as well with clear guidelines, something that is not done because of the lack of a range for the ssn :

[22] These template files and the processing pipeline code will be published on `http://ontology.irstea.fr`

[23] `http://jena.apache.org/documentation/serving\_data/soh.html`

observationResultTime property. We have seen examples in other datasets where the observation time is provided directly as an xsd:dateTime, whereas in our case we have preferred using the W3C Time Ontology so as to allow easier SPARQL queries when aiming at doing aggregations through the GROUP BY operator available in SPARQL.

In the coming months we will do similar processes to other weather stations owned by Irstea in other sites in France, so that they can be used by other researchers outside our institution in their research. We will also work with the association of quality indicators to data measured by the weather station or calculated by different processes over the initial data that have been captured.

References

1. Best practices for publishing linked data. http://www.w3.org/TR/ld-bp/. Accessed: 2014-07-08.
2. Climate and forecast (cf) features. `http://www.w3.org/2005/Incubator/ssn/ssnx/cf/cf-feature`. Accessed: 2014-06-30.
3. Isa programme location core vocabulary. `http://www.w3.org/ns/locn#`. Accessed: 2014-06-30.
4. Library for quantity kinds and units: schema, based on qudv model omg sysml(tm), version 1.2. `http://www.w3.org/2005/Incubator/ssn/ssnx/qu/qu`. Accessed: 2014-06-30.
5. Ogc geosparql - a geographic query language for rdf data. `http://www.opengeospatial.org/standards/geosparql`. Accessed: 2014-06-30.
6. Ontology for meteorological sensors. `http://www.w3.org/2005/Incubator/ssn/ssnx/meteo/aws`. Accessed: 2014-06-30.
7. Owl code of the time ontology. `http://www.w3.org/2006/time`. Accessed: 2014-06-30.
8. Semantic sensor network xg final report. `http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/`. Accessed: 2014-06-30.
9. G Atemezing, O Corcho, D Garijo, J Mora, Poveda-Villalon M., P Rozas, D Vila-Suero, and B Villazon-Terrazas. Transforming meteorological data into linked data. *Semantic Web journal, Special Issue on Linked Dataset descriptions, 2012. IOS Press, ISSN: 1570-0844*, 07 2012.
10. J-P Calbimonte, H Jeung, O Corcho, and K Aberer. Semantic sensor data search in a large-scale federated sensor network. In K Taylor, A Ayyagari, and D De Roure, editors, *SSN*, volume 839 of *CEUR Workshop Proceedings*, pages 23–38. CEUR-WS.org, 2011.
11. Jerry R. Hobbs and Feng Pan. Time ontology in owl. *W3C Working Draft*, 09 2006.
12. L. Lefort, J. Bobruk, A. Haller, K. Taylor, and A. Woolf. A linked sensor data cube for a 100 year homogenised daily temperature dataset. In *In 5th International Workshop on Semantic Sensor Networks (SSN-2012), CEUR-Proceedings*, 2012.
13. M. Stocker, Baranizadeh E., H Portin, M. Komppula, M. Rönkkö, A. Hamed, A. Virtanen, K. Lehtinen, A. Laaksonen, and M. Kolehmainen. Representing situational knowledge acquired from sensor data for atmospheric phenomena. *Environmental Modelling and Software*, 58:27–47, 2014.

measurement	feature of interest instance of $cf-feature$	property instance of $cf-property$
outdoor temperature	air	air_temperature
atmospheric pressure	air	air_pressure
humidity	air	relative_humidity
wind direction	wind	wind_from_direction
wind speed	wind	wind_speed
quantity of precipitation	rainfall	rainfall_amount
speed of precipitation	rainfall	rainfall_rate
solar radiation	ground_level_soil	downward_heat_flux_ at_ground_level_in_soil

Table 2. Weather measurement properties

Object	Resource class	local ID pattern
Station	$ssn:System$	⟨location Name⟩⟨ station type ⟩_⟨number ID⟩
Windwane	$ssn:Sensor$	⟨station ID⟩_⟨sensor type⟩_⟨number ID⟩
Instant	$time:Instant$	⟨date⟩T⟨time⟩⟨time zone⟩ (ISO 8601 format)
Interval	$time:Interval$	P⟨period⟩⟨unit⟩_⟨date⟩T⟨time⟩⟨time zone⟩
Observation	$ssn:Observation$	at_⟨timestamp⟩_of_⟨sensor⟩_on_⟨property⟩
	Examples	
URI station	`irstea:resource/system#lesPalaquinsVp2_01`	
URI windwane	`irstea:resource/sensor#lesPalaquinsVp2_01_WindVane_01`	
URI instant	`irstea:resource/instant#2013-01-23T13:30+0100`	
URI interval	`irstea:resource/interval#P30M_2013-01-23T13:30+0100`	
URI observation	`irstea:resource/observation#at_2012-06-19T15:45+0200_ of_lesPalaquinsVp2_01_Thermometer_01_on_AirTemperature`	

Table 3. URI generation templates for resources

Measured property	timestamp
exterior temperature	instant
atmospheric pressure	instant
air humidity	instant
wind direction	interval
wind speed	interval
rainfall amount	interval
rainfall speed	interval
solar radiation	interval

Table 4. Types of timestamps associated to the measured properties of our weather station

Demo Paper: Helping IoT Application Developers with Sensor-based Linked Open Rules

Amelie Gyrard, Christian Bonnet, and Karima Boudaoud

Eurecom, Sophia Antipolis, France
{gyrard,bonnet}@eurecom.fr
Laboratoire I3S-CNRS/UNSA, Sophia Antipolis, France
karima@polytech.unice.fr

Abstract. Domain-specific Internet of Things (IoT) applications are becoming more and more popular. They process data coming from sensor measurements. Adding semantic annotations to the sensory observations and measurements can allow to reason on data via logical rules. Stemming from Linked Open Data and Linked Open Vocabularies, we have designed sensor-based Linked Open Rules (S-LOR). S-LOR allows exploiting, reusing and combining rules to help developers design and combine cross-domain IoT applications. A proof-of-concept of S-LOR is available online at
http://www.sensormeasurement.appspot.com/?p=swot_template

Keywords: Semantic Web of Things (SWoT), Linked Open Rules, Linked Open Vocabularies, Sensor, Domain ontologies, Semantic Sensor Networks, Machine-to-Machine (M2M), Internet of Things (IoT), Semantic Web.

1 Introduction

Semantic annotations are applied to sensor in more than 200 ontology-based projects[1] in specific domains such as health care, smart home, tourism, transportation and agriculture. If all the semantic models and data representations for these works were published online, we would be able to reuse the smart reasoning methods to exploit, reuse and combine rules. If the rules are designed and implemented in a uniform way, it would be easy to automatically extract rules to build Sensor-based Linked Open Rules (S-LOR). As a first step, we built manually S-LOR to show its benefits: (1) rules are reused since they have already been designed and implemented by domain experts and (2) rules are interlinked with each other to combine domains to build smart IoT applications. For example, the rule `if foggy then switch on fog lamp` combines the weather domain thanks to the `foggy` concept and the transportation domain thanks to the `fog`

[1] http://www.sensormeasurement.appspot.com/?p=ontologies

`lamp` concept. Sheth et al. [1] and Wei et al. [2] are the early works that propose the idea to reason on semantic sensor data (e.g., to deduce potentially icy, blizzard, freezing concepts). Khandelwal et al. [3] propose Rule Interchange Format (RIF) as a standard format for the 'Linked Rules'. Seye el al. [4] implement a tool to convert RIF rules into SPARQL CONSTRUCT rules and design a RIF validator. RIF[2] is designed by the W3C to unify various rule languages: SWRL, RuleML (Rule Markup Language), R2ML (REWERSE Rule Markup Language), F-logic but we did not find any RIF-based tools that are already implemented to extract rules. None of these works propose to reuse and extract rules already implemented in ontology-based IoT projects. In this paper we present S-LOR to help developers integrate a smart reasoning in their IoT applications. Developers easily find rules related to sensor observations designed by domain experts which are represented in an unified way and are interoperable with each others since rules are designed according to the Machine-to-Machine (M3) ontology[3]. M3 is an extension of the W3C SSN[4] ObservationValue concept and describes sensors, measurements, units and domains to provide a basis for reasoning that can ease the development of advanced applications.

2 Sensor-based Linked Open Rules (S-LOR)

Automatic extraction of the linked rules is a challenging task due to heterogeneous terminologies or rule formats and syntaxes:

- Frequently, domain experts use popular ontology editor tools to design `owl:Restriction` rules. Unfortunately, these rules are not designed in the same way. In S-LOR, we convert them into SWRL, implemented with the Jena[5] framework and define an `owl:Restriction` template compatible with the M3 ontology.
- The rules are implemented with various ontology editors such as Protege, OWL API, SWOOPS, Hozo, TopBraid, OWL DL ed2, Neon but do not have the same syntax. Protege proposes at least 7 different plugins to write SWRL rules.
- The syntax of the rules is not the same according to the inference engine used, the rules cannot be inferred by another reasoner such as Jena, Pellet, Jess, Racer.
- The rules are not published online, we extract them manually from research articles (e.g., explanations, screenshots).
- Some rules are implemented with the SPARQL Inferencing Notation (SPIN)[6] language. SWRL rules can have equivalent rules written with SPIN language (SPARQL CONSTRUCT).

[2] http://www.w3.org/2005/rules/wiki/RIF_FAQ
[3] http://www.sensormeasurement.appspot.com/m3#
[4] http://www.w3.org/2005/Incubator/ssn/ssnx/ssn
[5] http://jena.apache.org/
[6] http://spinrdf.org/

– Different terms are used to describe sensor measurements and inferred concepts. We are working with ontology mapping tools such as LogMap and Aroma to automatically align concepts described in different ways such as etymology (e.g., rain/rainy), synonyms (e.g., precipitation/rain), different entities (e.g., driver's state defined as concepts or as properties).
– Redundant rules are not inserted in S-LOR, but we cite the author's work.
– Divergence in rules. We observe that if one work defines 16 rules related to wind speed and another work only 5 rules, we detect that the values are incompatible. The more rules are defined by domain experts, the more precise they will be. We update S-LOR according to the most precise work.

For these reasons, we manually extracted rules to build S-LOR. It would be possible to automate rule extraction if they are implemented as `owl:Restriction` with Protege or OWL API and following our M3 template.

To design a rule-based IoT application, developers get rules related to sensor data through S-LOR. S-LOR has been designed to be compliant with the Machine-to-Machine (M3) ontology [5] and the Jena inference engine[7]. The M3 ontology is an intermediary step while waiting for the adoption of better practices. By integrating M3, new IoT applications are becoming highly valued since domains are combined and rules reused. S-LOR has been used to design three IoT applications to link domains with each other: (1) transportation & weather[8] to suggest safety devices according to the weather (e.g., if snowy then safety devices are snow-chains, ABS, ESP), (2) tourism & weather[9] to suggest activities or clothes according to the weather, and (3) health care, weather & food with the naturopathy application[10] to suggest ingredients or recipes according to the user's emotional state, season, weather, etc. A sample of rules in S-LOR is depicted in Figure 1, `m3:Sensor` subclasses are displayed in the drop-down list. By choosing a sensor (e.g., `m3:Precipitation`), all rules related to this sensor are displayed such as `NoPrecipitation` or `HeavyRain` and the origins of the rules (e.g. Kofler, ThinkHome 2011 or Staroch 2013 as depicted in Figure 1). Rules are implemented according to the Jena rule syntax as following: `[HeavyRain: (?measurement rdf:type m3:Precipitation) (?measurement m3:hasValue ?v) (?measurement m3:hasUnit m3:MilimeterPerHour) greaterThan(?v,20) lessThan(?v,50) -> (?measurement rdf:type weather-dataset:HeavyRain) ]`

IoT developers can reuse the M3 ontology[11] and sensor-based Jena rules[12] published online. To evaluate S-LOR, an evaluation form has been set up[13] to be filled by IoT developers. This process is ongoing.

[7] http://jena.apache.org/documentation/inference/
[8] http://www.sensormeasurement.appspot.com/?p=transport
[9] http://www.sensormeasurement.appspot.com/?p=tourism
[10] http://www.sensormeasurement.appspot.com/?p=naturopathy
[11] http://www.sensormeasurement.appspot.com/m3#
[12] http://www.sensormeasurement.appspot.com/RULES/LinkedOpenRules.txt
[13] https://docs.google.com/forms/d/1HR2I4VbkHyAyKM1ElJp3bON-Y3kk94YP2cIQDnxdCPU/viewform

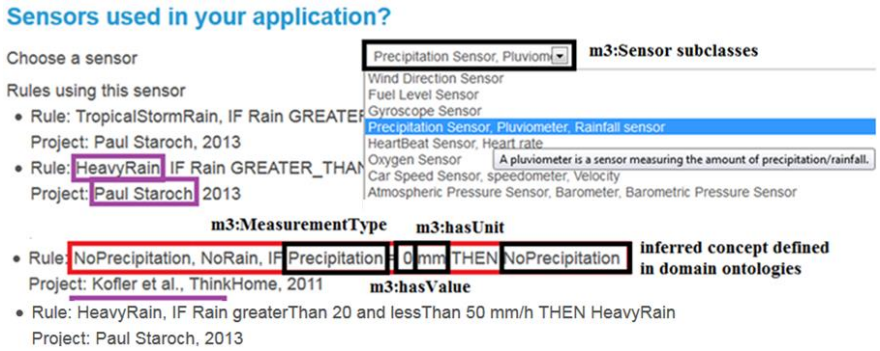


Fig. 1. Sensor-based Linked Open Rules

We designed Sensor-based Linked Open Rules (S-LOR) to extract, reuse and combine rules to help developers design smart rule-based IoT applications. S-LOR is integrated in our M3 framework to ease the reasoning on sensor data in IoT. Future works are to automatically extract rules and rewrite it to be compliant with M3 rules.

Acknowledgments. The authors would like to thank colleagues/friends/students for fruitful discussions, valuable feedback or help for the implementation. This work is supported by the Com4Innov Platform of Pole SCS[14].

References

1. Sheth, A., Henson, C., Sahoo, S.: Semantic sensor web. Internet Computing, IEEE **12**(4) (2008) 78–83
2. Wei, W., Barnaghi, P.: Semantic annotation and reasoning for sensor data. Smart Sensing and Context (2009) 66–76
3. Khandelwal, A., Jacobi, I., Kagal, L.: Linked rules: principles for rule reuse on the web. In: Web Reasoning and Rule Systems. Springer (2011) 108–123
4. Seye, O., Faron-Zucker, C., Corby, O., Follenfant, C.: Bridging the gap between rif and sparql: Implementation of a rif dialect with a sparql rule engine. AImWD 2012 (2012) 19
5. Gyrard, A., Bonnet, C., Boudaoud, K.: Enrich machine-to-machine data with semantic web technologies for cross-domain applications. In: WF-IOT 2014, World Forum on Internet of Things, 6-8 March 2014, Seoul, Korea, Seoul, KOREA, REPUBLIC OF (03 2014)

[14] http://www.pole-scs.org/

Demonstration: Web-based Visualisation and Monitoring of Smart Meters using CQELS

Maxim Kolchin, Dmitry Mouromtsev, Sergey Arustamov

ITMO University
Saint-Petersburg, Russia
kolchinmax@niuitmo.ru, {d.muromtsev,sergey.arustamov}@gmail.com

Abstract. In this demonstration paper we describe a web application that allows visualisation of current and historical readings of smart meters and monitoring their current states using CQELS for an integration of static and streaming RDF sources.

Keywords: Ontologies, RDF stream processing, Smart Meter, Data Integration, Visualisation

1 Introduction

Semantic Web technologies have already proved [1, 2] that they may become a tool for solutions aiming to solve issues with syntactic and semantic integration of data in heterogeneous formats, represented by different models. RDF and OWL provide a common model and vocabulary for such type of data with SPARQL to query data sources and Reasoning to infer "new information" using first-order predicate logic and inference rules. Moreover, RDF stream processing technologies allow combining background knowledge and streaming data together.

These sort of ideas may be applied with Smart Meter data where integration of static and streaming data is important to monitor effectively the state of the meters and the whole network and also respond immediately to emergency situations and unpredictable events.

In our case we have three types of smart electric meters and totally 55 instances that are able to transmit their readings each 5 minutes to a server. Our goal is twofold: (a) to be able to see the current reading in *real-time* and previous ones within a period with charts and (b) more importantly to automate identification of emergency situations, i.e too high voltage value in a line.

In this demo we show how we have tuned the web application to access streaming data coming from smart meters the same way it access a SPARQL endpoint.

The source code of the web application can be found on Github[1].

[1] The source code of the application, URL: `https://github.com/ailabitmo/daafse`

2 Architecture & Implementation

The architecture of the application is shown on Fig 1. The streaming data (the current readings) coming from smart meters is collected and transformed by RDF stream publisher to RDF streams that are in their turn published through the Message Bus implementing AMQP[2] protocol. Each RDF stream has an URI represented by AMQP URI[3] schema, i.e `amqp://example.com:10000/vhost?exchangeName=exchange&routingKey=mercury230_13534128`. The static data in SPARQL endpoint presents information about smart meters, like a type, serial number, installation location, precision, operating range and etc. Integration of static and streaming sources is performed through CQELS engine and an extension of SPARQL language supporting RDF streams. CQELS engine and the query language are presented in the paper [3].

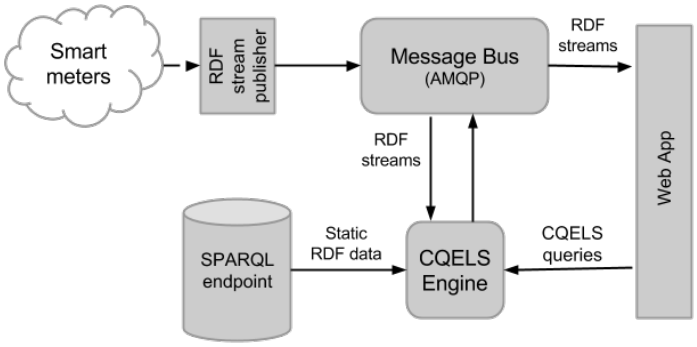


Fig. 1. System architecture: The App, Smart Meters, SPARQL endpoint and RDF streams

The Web App visualises the readings in a form of charts in *real-time* and allows to register a CQELS query that goes off alerts in case if there are some issues in functioning of smart meter or the network, i.e. there is a way to register a query the signals of a too high voltage value ($\geq 220V + 10\%$) measured at least by one of *known* smart meters.

The result of the query is a bag of triples representing an alert that is published to an RDF stream. See an example query and alert in Section 2.2.

[2] Advanced Message Queuing Protocol (AMQP), URL: `http://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol`

[3] AMQP URI spec, URL: `https://www.rabbitmq.com/uri-spec.html`

2.1 Ontologies

Several domain ontologies are used to represent the readings, smart meters, and alerts: Semantic Sensor Network (SSN)[4] ontology, the Electric Meters (EM)[5] ontology that is an extension of SSN ontology developed for this application and The DOLCE+DnS Ultralite ontology[6]. On Fig. 2 you can see a smart meter and an observation represented with these ontologies.

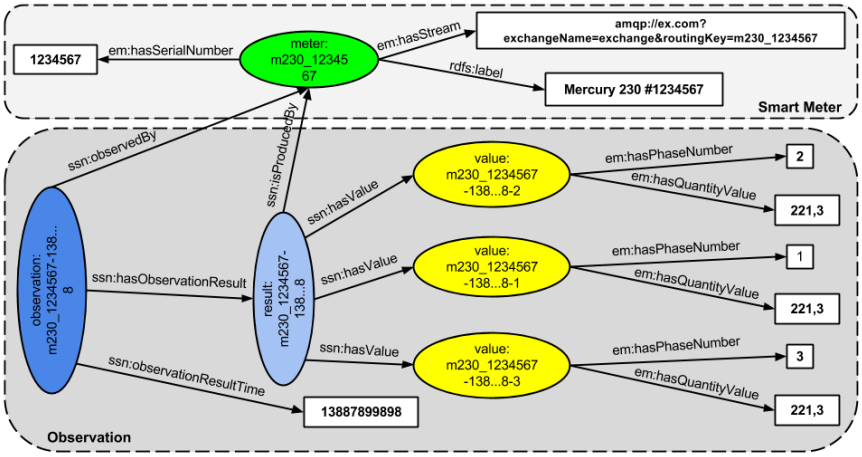


Fig. 2. Smart Meter and Observation

The EM ontology extends SSN ontology with several classes and properties:

- *em:ElectricityFeature* is a subclass of *ssn:FeatureofInterest*,
- *em:PolyphaseVoltageObservation* is a subclass of *ssn:Observation*,
- *em:hasStream* object property representing an RDF stream URI,
- and other classes and properties.

2.2 An example query

Below we show an example of query firing an alert each time when voltage value in any of electric phases measured by one of smart meters is $\geq 220V + 10\%$. All alerts are sent to a predefined RDF stream and consumed by the Web App.

[4] Semantic Sensor Network ontology, URL: `http://purl.oclc.org/NET/ssnx/ssn#`
[5] The Electric Meters ontology, URL: `http://purl.org/NET/ssnext/electricmeters#`
[6] DOLCE+DnS Ultralite ontology, URL: `http://www.loa-cnr.it/ontologies/DUL.owl#`

```
CONSTRUCT {
    ?alert a :TooHighVoltageValue ;
        dul:hasEventDate ?time ;
        dul:involvesAgent ?meter .
}
FROM NAMED <http://ex.com/sparql>
WHERE {
    GRAPH <http://ex.com/SmartMeters/> { ?meter em:hasStream ?stream }
    STREAM ?stream [NOW] {
        ?observation a em:PolyphaseVoltageObservation ;
            ssn:observationResultTime ?time .
        ?output a em:PolyphaseVoltageSensorOutput ;
            ssn:isProducedBy ?meter ;
            ssn:hasValue ?value .
        ?value em:hasQuantityValue ?qvalue .
    }
    FILTER(?qvalue >= (220 + 220*0.1))
    BIND (IRI(CONCAT(STR(?meter), '/alerts/', STR(?time))) AS ?alert)
}
```

RDF stream URIs of smart meters are queried from *http://ex.com/SmartMeters/* graph in *http://ex.com/sparql* SPARQL endpoint. In a more complicated scenario the highest possible voltage value for different smart meters may vary and instead of creating a separate alert for each type of meters this value can be queried from a SPARQL endpoint.

In this example an alert is an instance of *:TooHighVoltage* class that is a subclass of *dul:Event* class. Below a set of triples describing the alert:

```
:TooHighVoltage owl:subClassOf dul:Event ;
    rdfs:comment ...@en ;
    rdfs:label ...@en ;
    dul:isClassifiedBy :Warning .
```

Acknowledgments. This work has been partially financially supported by the Government of Russian Federation, Grant #074-U01.

References

1. Tallevi-Diotallevi S., Kotoulas S., Foschini L., Lcu F., Corradi A.: Real-Time Urban Monitoring in Dublin Using Semantic and Stream Technologies. In: The Semantic Web–ISWC 2013, pp. 178–194. Springer Berlin Heidelberg (2013)
2. Balduini M., Della Valle E., Dell'Aglio D., Tsytsarau M., Palpanas T., Confalonieri C.: Social listening of city scale events using the streaming linked data framework. In: The Semantic Web–ISWC 2013, pp. 1–16. Springer Berlin Heidelberg, (2013)
3. Le-Phuoc D., Dao-Tran M., Xavier Parreira J., Hauswirth M.: A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data. In: The Semantic Web ISWC 2011, pp. 370–388. Springer Berlin Heidelberg, (2011)