Linked Sensor Data:
RESTfully serving RDF and GML

Kevin R. Page[1], David C. De Roure[1], Kirk Martinez[1],
Jason D. Sadler[2], Oles Y. Kit[2]

[1] School of Electronics and Computer Science, University of Southampton, UK
[2] GeoData Institute, University of Southampton, UK

Abstract Publishing sensor observations on the Linked Data web is the first step in enabling the development of semantic web applications and mashups that can utilise sensor data. We present the design for a prototype API exposing data from the Channel Coastal Observatory in the UK. By combining REST and Linked Data principles we support both Semantic Web clients, OGC GML clients, and hybrid applications that can transpose between these, and other, representations.

1 Introduction

There have been numerous successful applications of semantics and Semantic Web technology to sensor networks. But what happens once we have this semantically enriched sensor data? Can the wider world access and make use of it? Can we combine sensor data with other data sources in a useful manner?

The Linking Open Data project reminds us of the *web* part of *Semantic Web* through extensive use of resolvable resources and the links between them. With this in mind, we aim to design a high-level API for semantic mashups and web applications based on three objectives:

1. to publish the sensor observations as linked data, enabling Semantic Web client applications that can combine these observations with other domain information retrieved from the linked data web (e.g. land use, transport).
2. to RESTfully publish sensor observations to clients that support existing GML schema.
3. to allow the development of hybrid clients which can transpose between linked data and GML (or other) representations, taking best advantage of both.

In this paper we first introduce the key principles and practises of both Linked Data and REST, before applying these principles to the RDF and GML resources and representations of our web service.

We then describe, through examples, an instantiation of the API design over observations from the Channel Coastal Observatory in the UK, before highlighting some of the problems with the current design and potential extensions to the API.

2 Scope

This paper describes experiences designing a web service to expose sensor data on the linked data web, and transposing between this semantic representation and existing data encoding standards. As such it is primarily an infrastructure piece, and while the use of ontologies is required for RDF representations of resources in our implementation, we do not cover specific ontologies - or the design of new ontologies - in any detail. As the service progresses toward public release long-term support for the ontologies we utilise will become more important; we believe the principles outlined here will be applicable as and when a more permanent and complete ontology set is selected. In the meantime such ontologies - both for sensor and observation concepts, and domain ontologies for features and observed properties - are under active development both within the EU *Semantic Sensor Grids for Rapid Application Development for Environmental Management* project (SemSorGrid4Env), and forums such as the W3C Semantic Sensor Networks Incubator Group.

3 Linked Data on the Semantic Web

> Linked Data refers to data published on the Web in such a way that it is machine-readable, its meaning is explicitly defined, it is linked to other external data sets, and can in turn be linked to from external data sets. [1]

Linked data can also be seen as an attempt to apply the defining characteristic of the document Web - the link - to a machine-readable, self-describing, web of data. Perhaps put another way, it's a reminder there is a 'Web' as well as a 'Semantic' in 'Semantic Web'!

The linked data web is constructed atop a number of principles and best practice. These were first set out by Berners-Lee in his 2006 design note[2]:

1. Use URIs as names for things
2. Use HTTP URIs so that people can look up those names
3. When someone looks up a URI, provide useful information, using the standards (RDF, SPARQL)
4. Include links to other URIs, so that they can discover more things

While the use of URIs is common throughout the Semantic Web - not least as the basic element of RDF - the requirement to use HTTP URIs sets Linked Data deployment apart. It is a departure from the use of URIs purely as unique identifiers within the graph; in Linked Data they are also a means of retrieving parts of the graph relevant to that resource - the URIs can be *dereferenced*.

This dual use of HTTP URIs does not, however, remove the need to distinguish between the two uses: a web client must be able to tell the difference between a URI representing the *person* Tim Berners-Lee (a *non-information*

resource) and a URI providing information *about* Tim Berners-Lee (an *information resource*); even if, in the linked data web, we dereference the former to retrieve the latter.

A web server communicates this distinction to the client through a combination of the HTTP `303 See Other` redirection code (referred to as the httpRange-14 solution) and content negotiation, i.e. returning different representations according to the HTTP `Accept` header set by the client [3] .

There are two general cases for this solution (figure 1):

1. if an information resource describing the non-information resource has multiple representations (e.g. in RDF and HTML) *of the same information* then the web server should first redirect the client (via a `303` response) to the intermediate information resource URI (indicating the move from a non-information resource to an information resource), and then use content negotiation to return the appropriate representation. The `Content-Location` header should be used to confirm the URI of this representation.
2. if the information resources describing the non-information resource contain *different information* depending on which representation is requested (e.g. the RDF representation contains different information to the HTML, not just a different representation), then the web server should redirect the client (via a `303` response) directly to the information resource appropriate to the requested content type, without an intermediate common information resource. The `Content-Location` header should be used to confirm the URI of the returned representation.

Our implementation of this mechanism for sensor observations is detailed in section 6.1.

Finally, it is noted that the use of resolvable (HTTP) URIs does not imply the encoding of semantics within a URI, and that the syntax used by a web server when returning a resource should not be interpreted as having such meaning. Apparent abstractions of the URI API (e.g. `http://example.com/<element>/<type>/<time>`) can not, and should not, be provided - certainly when describing the interaction with a client application. Use and manipulation of such an abstraction might provide a useful shortcut for developers looking to manually locate and trial resources; the use of such 'friendly' URIs that may encourage this misuse are not without merit when providing manageable endpoints for developers and end-users; but a linked data client should primarily access new resources via the links asserted within the (RDF) graph.

4 REST

Representational state transfer (REST) is a set of design principles which have been popularly and successfully adopted in many ('RESTful') web services, and is typically framed as an alternative to 'heavyweight' web services, including as the WS-* family.

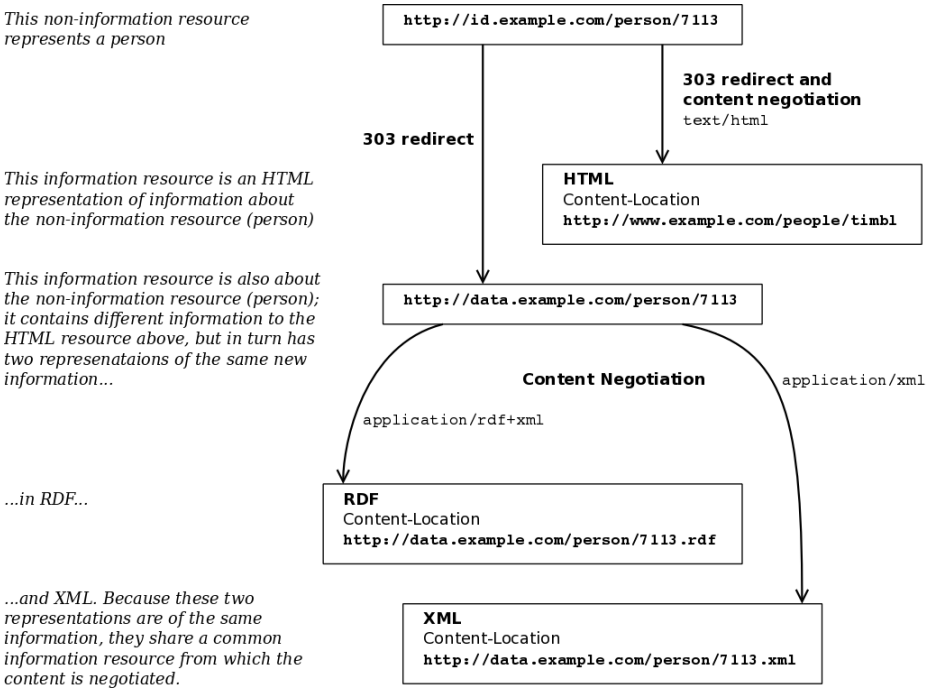

Figure 1. Dereferencing and content negotiation

REST aims to capture the features of the Web which have enabled it to scale so successfully:

- everything is a resource which is addressable
- resources have multiple representations
- relationships between resources are expressed through hyperlinks
- all resources share a common interface with a limited set of operations
- client server communication is stateless

REST is not, however, defined in terms of, nor limited to, the web [4] (though HTTP meets the REST criteria) and while there have been attempts to clarify the application of REST to web services through definition of a Resource Oriented Architecture [5] the term is still often loosely, sometimes incorrectly, applied.

The key principle of REST is the use of resources for specific things that we wish to reference, and the referencing of these resources using URIs. Representations of these resources – encoded in a particular format - are then accessed through the URI, usually using HTTP. REST limits the operations exposed by a web service to a small, well-defined, standard, set – for HTTP these are GET, POST, PUT, and DELETE, which are intrinsically compatible with (and a part of) Web architecture. This contrasts with a potentially expansive set of operators (for RPC style web services) or message contracts (for SOA style web services).

In the sense that they are both attempting to identify and replicate the successful architectural style and scalability of the Web, REST and Linked Data have shared aspirations; their focus on the use of resource URIs, Web style linking, and implementation using HTTP, provide a common technical bedrock which in, in both cases, leads to design strongly focused on the identification of resources and their representations.

As with Linked Data, URIs are both a retrieval mechanism and identifier; again the URI is opaque, and a client must access resources through the links provided from one resource representation to another.

While in many respects the Linked Data approach would appear to be RESTful, differences are found in the nature of the representations returned when resolving, or dereferencing, a resource URI. This would primarily seem to be an artifact of the data models typically associated with each approach: XML and RDF. XML has a design based on the concept of a document, so information about a resource is more easily and portably expressed as self-contained XML 'documents'; representations of resources in REST systems often follow this pattern (with links to other XML 'documents' describing other resources). RDF, on the other hand, allows – encourages – assertions about a resource to be made anywhere on the semantic web; it is therefore much less likely that dereferencing a resource will return the complete set of assertions about that resource.

5 Interfaces to Sensor Data

'Mashups' have demonstrated the practicality and popularity of APIs[3] (many RESTful) that enable the rapid development of web applications combining data from several sources. This data rarely shares a common data model and is normally combined 'manually' by the mashup developer; we propose the use of Linked Data to support more powerful and flexible web applications that can semantically utilise sensor data.

Exposing sensor data according to linked data principles and practice is a first and necessary step to enabling linked data applications. In this paper we introduce a dual purpose API design that combines the provision of linked sensor data with a read-only[4] RESTful interface to existing standardised data encodings. Using this API a client application can move seamlessly between RDF and GML representations, taking advantage of the semantic linking provided in linked data, while being able to retrieve established encodings for Web GIS applications when required. Conversely an application can use a GML identifier as a jumping off point into the linked data web.

GML[6] is a particularly interesting XML representation since it has several RDF-like features: an object-property-value model similar to the RDF model,

[3] http://www.programmableweb.com/

[4] The current design and implementation provides a read-only API; some would assert this does not constitute a complete RESTful service, however we contend that the design centred around resources and representations strongly conforms to the REST architectural style.

and extensive (if perhaps under-utilised) support for remote properties using `xlink:href`. This probably shouldn't come as a surprise: early versions of GML included an RDFS profile [7].

We take a data-(consumer-)centric approach to the sensor data and adopt the Observations and Measurements (O&M) model [8] through both its GML Application Schema and, by including some key concepts (Observation, ObservationCollection, Feature) and properties (samplingTime, resultTime, featureOfInterest, result, observedProperty, member), in a small ontology. While ontological evaluation has identified weaknesses in the O&M model[9], our primary focus – as noted in section 2 – is on the mechanisms for linking data and representations. A very simple ontology (figure 2) closely matching the O&M specification is sufficient for these preliminary prototypes pending adoption of more sophisticated ontologies for observations, devices, and domain data.

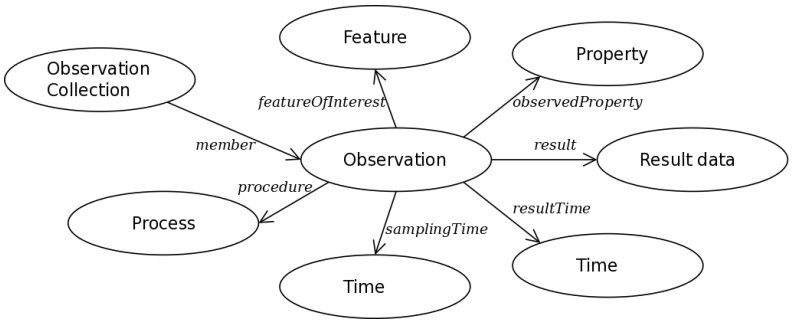

Figure 2. Key concepts and properties in the simple observation ontology

We also note that the O&M GML representation of resources (and SensorML for appropriate resources) is compatible with the O&M GML and SensorML returned by the Sensor Observation Service (SOS)[10] (particularly the GetObservation and DescribeSensor functions). SOS has been shown to offer advantages over WFS when publishing time series data[11]. The design described herein can also be considered a partial implementation of a RESTful SOS; we hope that this will allow adaption of SOS clients to work with our API.

6 Channel Coastal Observatory implementation

The Channel Coastal Observatory (CCO) is the data management centre for the Regional Coastal Monitoring Programmes of England. Over a period of more than 5 years, the GeoData Institute has designed, built from the top down, and operated the data management infrastructure to run this programme. This includes software to manage and transmit real-time data from the largest network of coastal sensors in the UK; a data management infrastructure to manage data

and metadata for over 65,000 environmental surveys of different types amounting to terabytes of storage; and a website to deliver real time and surveyed data to a public audience though highly complex dynamic map and data visualisation interfaces, serving over a million hits per month.

As part of the SemSorGrid4Env project a high-level API for web applications is being developed. This semantically exposes the CCO sensor network data, combining the principles and practice of REST APIs and linked data, enabling rapidly development of applications that will link sensor observations with diverse domain data (to support use cases such as flood planning and emergency response).

Initial development of the API has focused on publishing data from a network of marine and coastal sensors monitoring:

- wave height
- sea surface temperature
- wave period
- wave spread
- wave direction
- tide height

In this section we explore the current design of the prototype API by way of example.

6.1 Dereferencing and Content Negotiation

The (non-information) resource:

`http://id.channelcoast.org/observations/boscombe/Hs/20090801`

is the set of all observations of Hs (significant wave height) from the Boscombe sensor on August 1st 2009[5].

When a client attempts to dereference this resource (e.g. through an HTTP GET), the web server responds with code `303 See Other` and the information resource (figure 3):

`http://data.channelcoast.org/observations/boscombe/Hs/20090801`

Content negotiation is then used to retrieve a suitable representation:

- `application/rdf+xml` returns an RDF representation.
- `application/xml` returns a GML representation.
- `text/html` returns an HTML rendering for viewing in a traditional web browser.

[5] As noted in earlier sections, the URI for the resource should be treated as an opaque string; while the implied structure within the URI is helpful when designing and maintaining the web service (and perhaps for developers when writing clients), client applications must navigate to and between resources using links between those resources.

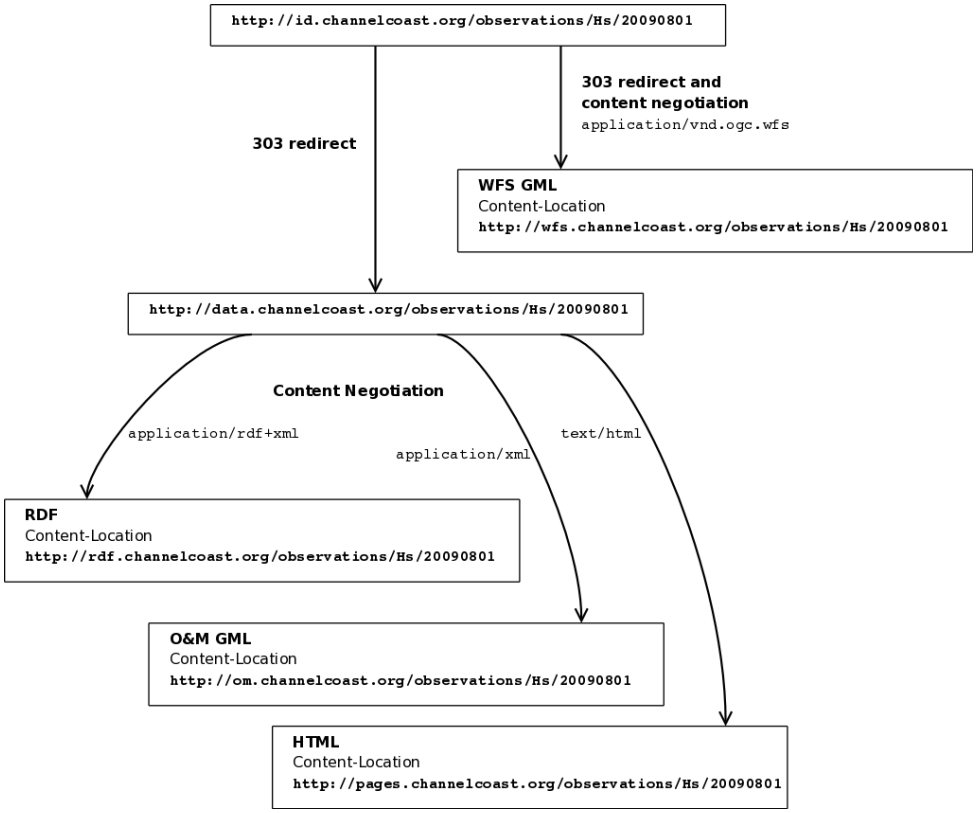

Figure 3. Dereferencing URIs on the CCO server

In each case the web server responds with code `200 OK` and sets the `Content-Location` header to the resource of the negotiated representation, e.g.

`http://rdf.channelcoast.org/observations/boscombe/Hs/20090801`

`http://om.channelcoast.org/observations/boscombe/Hs/20090801`

`http://pages.channelcoast.org/observations/boscombe/Hs/20090801`

followed by the appropriate representation in the HTTP body.

The intermediate stage of redirecting to a common information resource URI indicates to the client that the following content negotiation is for different representations of the *same* information. For example, this means that if the client has reached the resource through RDF representations, but needs to plot the data using an OGC compliant tool (e.g. within a mapping layer) it can request the GML representation knowing it is plotting the *same* information.

Other representations might, by necessity of the encoding, be of closely related but different information. An earlier incarnation of the CCO server implementation returned a GML representation using the WFS schema to create a MapServer compliant layer, and while we wish to preserve this functionality,

the 'flattened' nature of the data in the WFS layer means this representation contains different information to those based on O&M (described below).

In this situation we use the content type `application/vnd.ogc.wfs` to enable a client to retrieve a backwards-compatible representation; when the client requests the non-information resource:

`http://id.channelcoast.org/observations/boscombe/Hs/20090801`

with content type `application/vnd.ogc.wfs`, the server performs a `303` redirect directly to the WFS GML (setting the `Content-Location` header accordingly):

`http://wfs.channelcoast.org/observations/boscombe/Hs/20090801`

In redirecting directly to the WFS information resource through content negotiation (rather than through the common information resource and *then* performing content negotiation), the web server has indicated that this is a *different* (but related) information resource, not a different representation of the same information resource.

6.2 Identification of resources and representations

Given the consumer-centric[8] nature of the applications the CCO web service is expected to support, and a data set consisting principally of measurements, we have adopted the Observations and Measurements model when returning structured data to the client.

Figure 4 shows a number of relationships between key URIs, again focusing on the interface for retrieving significant wave height. Resources for other observed properties follow a similar structure, and examples of how these relationships are encoded in specific representations follows in section 6.3.

Since we wish to return XML (GML) representations as well as RDF, the identification and structuring of resources bears many similarities to the document (XML) oriented design of REST APIs (section 5): identification and structuring of resources can be very dependent on the data (the resources) being exposed. For example, our primary observation resources are of the form:

`http://id.channelcoast.org/observations/boscombe/Hs/20090801#140500`

where the individual observation is dereferenced by retrieving the resource:

`http://id.channelcoast.org/observations/boscombe/Hs/20090801`

This strikes a balance between the size of the retrieved resource representation and the number of links the client must retrieve *for this particular data set.* In this case the observations of Hs at Boscombe at taken half-hourly, so the resource that must be retrieved to dereference any one observation will contain 48 observations (all the observations for the day).

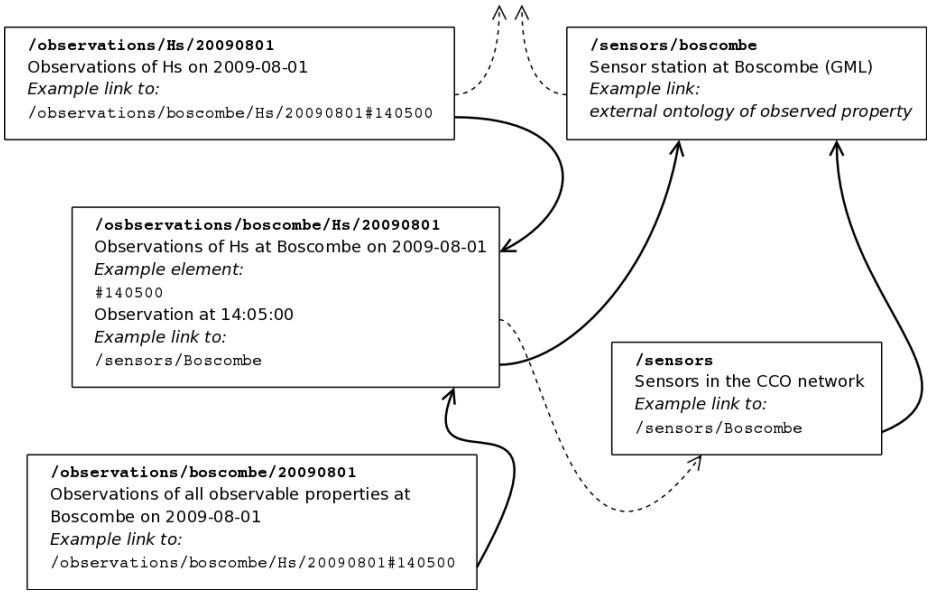

Figure 4. Selected resources and links from the `http://id.channelcoast.org/` namespace (not comprehensive).

6.3 Examples of resource representations

The following XML fragments demonstrate the content of the O&M GML and RDF representations of:

`http://id.channelcoast.org/observations/boscombe/Hs/20090801`

The O&M GML encoding is returned for `application/xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<om:ObservationCollection gml:id="this"
  xmlns:om="http://www.opengis.net/om/1.0"
  xmlns:gml="http://www.opengis.net/gml"
[...] >
  <gml:description>Wave height observations at Boscombe on 2009-08-01
      </gml:description>
  <om:member>
    <om:Observation gml:id="140500">
      <om:resultTime>
        <gml:TimeInstant gml:id="T140500">
          <gml:timePosition>2009-08-01T14:05:00</gml:timePosition>
        </gml:TimeInstant>
      </om:samplingTime>
      <om:samplingTime>
[...]
      </om:samplingTime>

      <om:procedure xlink:href=
          "http://id.channelcoast.org/sensors/boscombe"/>
      <om:observedProperty xlink:href=
          "http://marinemetadata.org/2005/08/ndbc_waves#Wind_Wave_Height"/>
      <om:featureOfInterest xlink:href=
```

```
          "http://www.eionet.europa.eu/gemet/concept?cp=7495"/>
      <om:result xsi:type="gml:MeasureType" uom="urn:ogc:def:uom:OGC:m">
        0.28
      </om:result>
    </om:Observation>
  </om:member>
  <om:member>
    <om:Observation gml:id="143500">
[...]
    </om:Observation>
  </om:member>
</om:ObservationCollection
```

The RDF representation is returned for `application/rdf+xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
  xmlns:om2="http://rdf.channelcoast.org/ontology/om_tmp.owl#"
[...]
>
  <rdf:Description rdf:about=
      "http://rdf.channelcoast.org/observations/boscombe/Hs/20090801">
    <rdfs:label>
      An RDF representation of wave height observations at Boscombe on 2009-08-01
    </rdfs:label>
  </rdf:Description>
  <om2:ObservationCollection
      rdf:about="http://id.channelcoast.org/observations/boscombe/Hs/20090801">
    <om2:member>
      <om2:Observation rdf:about=
          "http://id.channelcoast.org/observations/boscombe/Hs/20090801#140500">
        <om2:resultTime>
          <om2:TimeInstant rdf:about=
            "http://id.channelcoast.org/observations/boscombe/Hs/20090801#T140500">
[...]
          </om2:TimeInstant>
        </om2:resultTime>
[...]
        <om2:procedure rdf:resource="http://id.channelcoast.org/sensors/boscombe"/>
        <om2:observedProperty rdf:resource=
            "http://marinemetadata.org/2005/08/ndbc_waves#Wind_Wave_Height"/>
        <om2:featureOfInterest rdf:resource=
            "http://www.eionet.europa.eu/gemet/concept?cp=7495"/>
        <om2:result [...]
[...]
      </om2:Observation>
    </om2:member>
  </om2:ObservationCollection
</rdf:RDF>
```

While both excerpts have been reduced for brevity, the key point to note is that the URIs for resources are consistent between representations: when a client, having received a representation, dereferences one of the resources within, that URI will in turn be dereferenced according to the needs of the client application. The client can continue navigating through the same dimension of information space in RDF or GML, or transpose from one to the other.

Consistent use of URIs is maintained in other observation collections. For example:

`http://id.channelcoast.org/observations/Hs/20090801`

contains parallel fragments in GML:

```
<om:ObservationCollection gml:id="this" [...]>
[...]
<om:member>
  <om:Observation xlink:href=
      "http://id.channelcoast.org/observations/boscombe/Hs/20090801#140500"/>
[...]
```

and in RDF:

```
<o2:ObservationCollection rdf:about=
    "http://id.channelcoast.org/observations/Hs/20090801">
[...]
<om2:member rdf:resource=
    "http://id.channelcoast.org/observations/boscombe/Hs/20090801#140500"/>
[...]
```

7 Discussion

7.1 Limitations of combining GML and Linked Data

While there are clear benefits to alternately returning GML and Linked Data representations of a resource, each of these representations has particular constraints – be that conceptual model, design principle, or XML serialisation – and though they are in general complementary, a few specific interactions create as yet unresolved limitations of our approach.

With regard to aggregation and nesting of observation collections, the O&M GML Application Schema is relatively constrained: more specialised observation collections have typed constituents that are too specific to be transcluded in more general, otherwise specialised, or multiply nested, collections. While this is not an issue in existing O&M applications (e.g. using an SOS), the Linked Data principles lead us to both uniquely identify individual observations (i.e. avoiding duplication of a single observation at several URIs and creating duplication in the graph) and the identification and publication of aggregate resources that by their nature include observations that have already been 'used'/published as, or as a part of, another resource (i.e. we must use linking rather than duplication).

For example, it would be desirable to have the URI:

`http://id.channelcoast.org/observations/boscombe/Hs`

represent all the observations of Hs at Boscombe, and this to be an aggregation in the form of an ObservationCollection, where each member is in turn an ObservationCollection such as:

`http://id.channelcoast.org/observations/boscombe/Hs/20090801`

(there would be as many references to other ObservationCollections as there are days on which observations have been made).

While this is relatively simple to implement in RDF - `ObservationCollection` as a `subClass` of `Observation`, and `member` a `TransitiveProperty` with `rdfs:range`

`Observation` - this is not possible using the O&M GML Schema [12]. This leaves two possibilities for the O&M GML representation:

1. Do not return a GML representation and return a `406 Not Acceptable` code.
2. `303` redirect straight to an information resource content negotiated for `application/xml`, without a common information resource shared with the RDF representation. The body returned contains a 'flattened' ObservationCollection with xlinks directly to the Observations required; this is the similar to the approach taken for WFS compatibility (section 6.1).

While the current CCO design takes the latter approach, without nested aggregations the number of Observations returned is potentially very high, and as such a compromise is made to redefine the resource as e.g. Observations of Hs at Boscombe over the last week (rather than all time). A hybrid approach might combine the original resource definition with a `406` for GML with a second resource limited to collections that are reasonable to return in GML (e.g. the observations from the last week).

Further limitations of the O&M GML Schema, as documented in [12], restrict the expressive completeness of our representations. For example, the resource:

`http://id.channelcoast.org/observations/boscombe/Hs/20090801`

is a time series observation. However, if the O&M GML returned encoded this explicitly, the results would be within a `CV_DiscreteTimeInstantCoverage`. This typing would, in turn, limit linking to constituent observations. As with the previous example, these limitations can be overcome in the RDF representation.

8 Conclusions and future work

The web of Linked Data holds great potential for the creation of semantic applications that can combine self-describing structured data from many sources including sensor networks. These applications build upon the success of an earlier generation of 'rapidly developed' applications that used RESTful APIs, although the data really was 'mashed up' without any underlying common data model or principled integration.

Before linked data applications can use sensor observations, they must be published in a form compatible with the Linked Data principles. In this paper we have introduced one such API for the Channel Coastal Observatory, and shown how the consistent use of URIs across representations can also enable clients using XML representations such as GML, and hybrid clients that can switch between the two. In the next stages of the SemSorGrid4Env project we will be making use of the API to develop such applications in support of our flood and fire management use cases. Revision and iterative improvement of the API will doubtless follow based on this experience!

A notable omission from the current design is a form of query interface. As discussed in the previous section the number of members in a time series observation is potentially very large. While, from the position of publisher, one

can carefully allocate resources that return a 'reasonable' number of observations (as we did), this is unlikely to satisfy the needs of all users and developers. Given the potentially infinite combination of temporal ranges, while URIs could be speculatively allocated to represent these, the question then becomes how this great number of range resources are published and linked to (without the URI being overloaded as a query syntax and hard-coded into the system). While a SPARQL endpoint is not a requirement for publishing Linked Data (indeed several notable examples publish their linked data for a third party to provide a SPARQL endpoint for) some form of query interface will be considered for a future version of the CCO interface.

Acknowledgements

Thanks are due to Homme Zwaagstra for his work on earlier prototypes of the CCO REST interface; Juan Sequeda and Oscar Corcho for lively discussions on the nature of query interfaces for linked sensor data. The work in this paper was funded by the IST STREP Programme of the Commission of the European Communities as project number FP7-223913 "SemSorGrid4Env: Semantic Sensor Grids for Rapid Application Development for Environmental Management".

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked Data - The Story So Far. Special Issue on Linked Data, International Journal on Semantic Web and Information Systems (IJSWIS) (2009)
2. Berners-Lee, T.: Linked Data, Design Issues. http://www.w3.org/DesignIssues/LinkedData.html (27 July 2006)
3. Sauermann, L., Cyganiak, R.: Cool URIs for the Semantic Web. W3C Semantic Web Education and Outreach Interest Group Note (31 March 2008)
4. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis, Information and Computer Science, University of California, Irvine, California, USA (2000)
5. Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly & Associates (May 2007)
6. Portele, C.: OpenGIS Geography Markup Language (GML) Encoding Standard (OpenGIS Standard OGC 07-036). Technical report, Open Geospatial Consortium Inc. (27 August 2007)
7. Lake, R., Cuthbert, A.: Geography Markup Language (GML) v1.0. Technical report, OpenGIS Consortium (12 May 2000)
8. Cox, S.: Observations and Measurements - Part 1 - Observation schema (OpenGIS Implementation Standard OGC 07-022r1). Technical report, Open Geospatial Consortium Inc. (8 December 2007)
9. Probst, F.: Ontological Analysis of Observations and Measurements. In: Geographic Information Science, 4th International Conference (GIScience 2006). Number 4197 in Lecture Notes in Computer Science (2006) 304–320
10. Na, A., Priest, M.: Sensor Observation Service (OpenGIS Implementation Standard OGC 06-009r6). Technical report, Open Geospatial Consortium Inc. (26 October 2007)

11. Bermudez, L., Bogden, P., Bridger, E., Cook, T., Galvarino, C., Creager, G., Forrest, D., Graybeal, J.: Web feature service (WFS) and sensor observation service (SOS) comparison to publish time series data. In: 2009 International Symposium on Collaborative Technologies and Systems. (2009)
12. Henson, C., Neuhaus, H., Sheth, A., Thirunarayan, K., Buyya, R.: An Ontological Representation of Time Series Observations on the Semantic Sensor Web. Proceedings of 1st International Workshop on the Semantic Sensor Web 2009 at the 6th European Semantic Web Conference (ESWC 2009) (1 June 2009)