Epimorphics Ltd
Linked data solutions

INPSIRE in RDF

Deliverable 1

Version 0.5
19th May 2014

Changelog

Version	Date	Editor	Notes
0.1	27/03/2014	Stuart Williams	Initial draft.
0.2	11/04/2014	Stuart Williams	Added treatment of Hydrograph schema
0.3	22/04/2014	Stuart Williams	Extended post face-to-face workshop meeting. Added section on transforming data (4.2.2) in response to a comment from Michael Lutz Added section (5.7) on how the derived vocabularies could be adapted for use using an objects as graphs (see 3.2.1) approach. Added more diagrams.
0.4	24/04/2014	Stuart Williams	Added more diagrams and a little extra narrative in response to comments from Diederik Tirry.
0.5	19/05/2014	Stuart Williams	Added diagram for "Objects as Nodes and Graphs". Improved alignment of embedded turtle and diagrams.

INPSIRE in RDF

Contents

1 Introduction

This report has been prepared as input to the Are3na project activity to "Documentation and development of a methodology for INSPIRE in RDF".

This report discusses:

- Conceptual model clashes between UML/INSPIRE and RDF (Section 3)
- The emerging approach to persistent URI patterns being developed in the UK with some example from recent projects. (Section 4)
- Presentation of transformed INSPIRE application schema including fragments (sigificant fragments in some cases) from: (Section 5)
    - The INSPIRE Generic Conceptual Model (GCM)
    - Area Management Theme
    - Environmental Facilities Theme
    - Hydrography Theme
    - Geographical Names Theme
- We also present local UK extensions/specialisation of the Water Framework Directive application schema within the Area Management theme to include more detailed information held in the UK for catchment-planning. (Section 5.4)

The report closes with some concluding remarks on lingering issues.

2 References

[1] "INSPIRE Data Specification for the spatial data theme Environmental Monitoring Facilities", Version 3.0, European Commission Joint Research Centre, http://inspire.jrc.ec.europa.eu/documents/Data_Specifications/INSPIRE_DataSpecification_EF_v3.0.pdf

[2] "FOAF Vocabulary Specification", Dan Brickley, Libby Miller, http://xmlns.com/foaf/spec/

[3] "The Organization Ontology", W3C, Editor Dave Reynolds. http://www.w3.org/TR/vocab-org/

[4] "ISA Programme Location Core Vocabulary", EU ISA Programme Core Vocabularies Working Group (Location Task Force), http://www.w3.org/ns/locn.html

[5] "ISA Programme Person CoreVocabulary", EU ISA Programme Core Vocabularies Working Group (Location Task Force), http://www.w3.org/ns/person

[6] "RDF 1.1 TriG RDF Dataset Language", W3C, http://www.w3.org/TR/trig/

[7] "URLs in Data Primer", W3C TAG, Editor, Jeni Tennison, http://www.w3.org/TR/urls-in-data/

[8] "A Reusable Ontology for Fluents in OWL", Welty, Chris, Richard Fikes, and Selene Makarios., In *FOIS*, vol. 150, pp. 226-236. 2006. http://www.comp.leeds.ac.uk/brandon/FOIS-06/CRC/Part-5/20_fois06.pdf

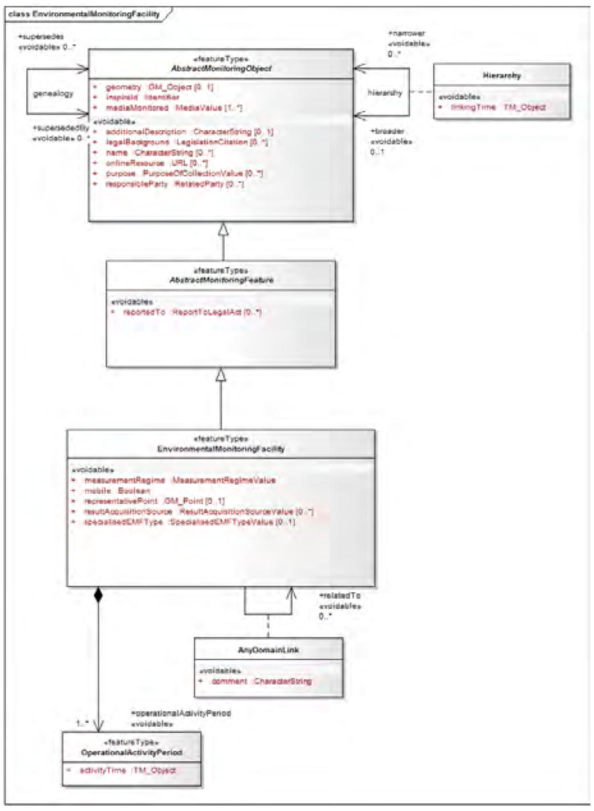
[9] " Designing URI Sets for Location", CTO Office and UK Location Programme, May 2011. http://data.gov.uk/sites/default/files/Designing_URI_Sets_for_Location-V1.0_10.pdf

[10] "URI Patterns" Draft revision to UK URI pattern guidance from UK Gov Linked Data WG #UKGovLD, editor Stuart Williams, http://tinyurl.com/UKGovLD-revisedUriPatterns

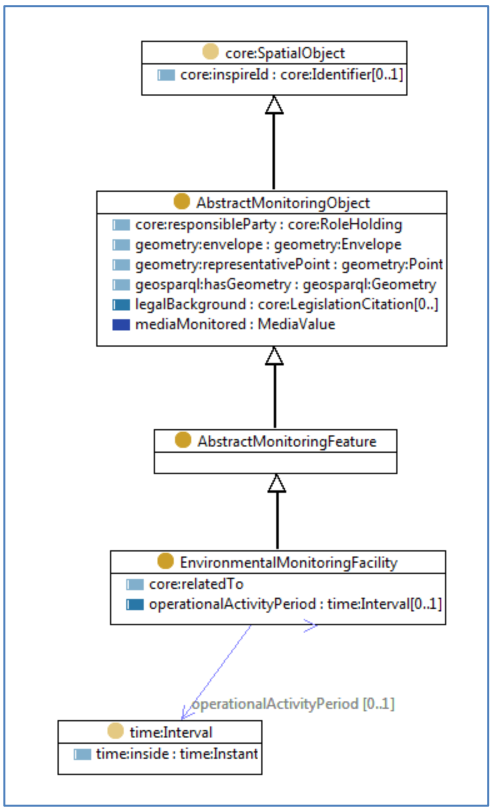
[11] "URI for Location" Draft revision to UK URI pattern guidance UK Gov Linked Data WG #UKGovLD, editor Stuart Williams, http://tinyurl.com/uri-for-location

3 INSPIRE v RDF Conceptual Model Clashes

The availability of visual tools for developing both UML and RDF data models can lead one to think that there is a relatively small gap between their two world views. For example the following diagrams are related to the Environmental Monitoring Facilities data specification [1]



class EnvironmentalMonitoringFacility
+supersedes «voidable» 0..*
genealogy
+supersededBy «voidable» 0..*
«featureType» AbstractMonitoringObject
+ geometry : GM_Object [0..1]
+ inspireId : Identifier
+ mediaMonitored : MediaValue [1..*]
«voidable»
+ additionalDescription : CharacterString [0..1]
+ legalBackground : LegislationCitation [0..*]
+ name : CharacterString [0..*]
+ onlineResource : URL [0..*]
+ purpose : PurposeOfCollectionValue [0..*]
+ responsibleParty : RelatedParty [0..*]
+narrower «voidable» 0..*
hierarchy
+broader «voidable» 0..1
Hierarchy
«voidable»
+ linkingTime : TM_Object
«featureType» AbstractMonitoringFeature
«voidable»
+ reportedTo : ReportToLegalAct [0..*]
«featureType» EnvironmentalMonitoringFacility
«voidable»
+ measurementRegime : MeasurementRegimeValue
+ mobile : Boolean
+ representativePoint : GM_Point [0..1]
+ resultAcquisitionSource : ResultAcquisitionSourceValue [0..*]
+ specialisedEMFType : SpecialisedEMFTypeValue [0..1]
+relatedTo «voidable» 0..*
AnyDomainLink
«voidable»
+ comment : CharacterString
1..*
+operationalActivityPeriod «voidable»
«featureType» OperationalActivityPeriod
+ activityTime : TM_Object



core:SpatialObject
core:inspireId : core:Identifier[0..1]
AbstractMonitoringObject
core:responsibleParty : core:RoleHolding
geometry:envelope : geometry:Envelope
geometry:representativePoint : geometry:Point
geosparql:hasGeometry : geosparql:Geometry
legalBackground : core:LegislationCitation[0..]
mediaMonitored : MediaValue
AbstractMonitoringFeature
EnvironmentalMonitoringFacility
core:relatedTo
operationalActivityPeriod : time:Interval[0..1]
operationalActivityPeriod [0..1]
time:Interval
time:inside : time:Instant

The diagram of the left is taken from the Environmental Monitoring Facilities application schema contained within the INSPIRE consolidated UML model available at http://inspire.jrc.ec.europa.eu/data-models/ view using the Sparx Enterprise Architect tool. The diagram on the left was created using the Topbraid Composer, an RDF/OWL ontology editing tool and workbench.

However, despite the visual similarity of the diagrams there are two significant differences in the underlying models that is worthwhile considering.

- scoping of properties
- what statements are being implicitly/explicitly made about what 'things'?

3.1 Scope of properties

In UML properties (attribute and associations or more specifically association roles) are scoped to the UML Classes in which they are defined and inherited by subclasses thereof. By "scoped" we mean that the definition of a named property is determined by the UML Class with which it is associated. Two independent UML classes may each bear attributes with have the name, say, *'label'* and the purpose or significance of the attribute may be identical, however, strictly, because the two classes are independent, they are two different UML attributes. As we will see later, section 4.2.1, approaches to generating RDF property URI for UML attributes or association-roles tend to do so by syntactically prefixing the given UML properties 'local' name with tokens that name the associated UML class and containing application schema. That the resulting URI are distinct further emphasises the distinct nature of the attributes/association roles designated by the URI. While in this example we have attributed common significance or purpose to these two properties, in general UML attributes/association-roles on independent UML classes whose names are identical may have completely different definitions and purpose.

Put simply, in UML, properties/association roles are defined in relative to the UML classes that 'use' them. They do not and cannot have an independent existence.

In contrast, in RDF properties are first class entities that can exist independently of the classes that use them. They may be 'bound' to RDF/OWL classes simply through the use of '*rdfs:domain*' statements or more complexly (in OWL) through the use of property restrictions that can impose cardinality restrictions on the use of the property[1]. It is also possible to specialise the range of a property used in conjunction with instances of a given RDF/OWL class - although such specialisations may only narrow rather than extend the range of a property.

In RDF/OWL (in the presence of reasoners) the class of an entity (designated by a URI or a blank node) may be established by reasoning over its properties and their values. That is to say that the RDF/OWL class (or indeed classes) of a 'thing' can follow from its properties - whereas in UML the class of an instance very much restricts the properties that it may bear, except by specialisations (sub-classes) that extend or override the properties that may be used with the extended class.

The point being made here is that in UML the properties that can used with a UML class follow from the class definition, whereas in RDF/OWL the RDF/OWL class of a thing, whether or not stated explicitly can follow from the statements (properties and values) made about it.

Because of the central role of properties, more so than classes, in comprehending RDF statements there are strong cultural imperatives to reuse and share existing terminology rather than 're-invent'. So for example, "*skos:prefLabel*", "*skos:altLabel*" and "*rdfs:label*" are widely used (annotation) properties for labelling entity. The first two are sub properties of the third such that whether explicity stated or not any RDF statements made using either of the SKOS labelling properties entails RDF statements made using the RDFS property:

```
{ ?s skos:prefLabel ?label } => {?s rdfs:label ?label }
```

Where more specialised properties are created, with perhaps more nuanced meaning, a common pattern is to define them as a sub-property of a more general property so that a least some level of partial 'understanding' can be achieved - based on a wider understanding of the more general property.

In order to make RDF/OWL properties available for wide reuse, a common practice is to leave their domains (the set of subject entities that bear the property) as open as possible, possible as fully open as '*rdfs:Resource'* (which is the default if nothing more restrictive is stated) or *'owl:Thing'*. This leaves a property available to be applied to more or less anything at all. As noted above, the use of a property with a given class can be restricted by the class definition.

It is less common to leave a property range (the set of entities to which a property may refer) open. OWL, more so than RDFS, makes a distinction between datatype (literal valued) and object (URI or blank node valued) properties.

[**Insert: 2014/04-17**
In discussion at the F2F meeting on 15th April, Clemens remarked on this when suggesting more or

[1] Strictly cardinality in RDF/OWL cardinality restrictions are on the number of entities designated by the object (s,p,o sense of object) of statements made using a property and not on the number of times the property may be used. In RDF/OWLs open-world entities may be identified by more than one URI (no unique name assumption) so it is not necessarily invalid/inconsistent to make say 3 statements using a property that has a cardinality restriction of 2 with respect to a given class because two or more object URI may designate the same 'thing'. Similarly, it is not invalid/inconsistent in RDF/OWL to make fewer statements with a given property than a minimum cardinality restriction requires. A person may have two parents, but we may only be capable of stating one of them - that that is so does not alter that a person has two parent (that are also people).

less the complete removal of domain, range and cardinality constraints from the RDF representation or at least opening ranges as wide as reasonably possible.

If we adopt an objects as graphs approach, we either have to:

- let the spatial-object types be mixin types that apply to the abstracted things rather than spatial-objects
- create a parallel set of real-world classes for the things abstracted by given spatial-object type.
- leave property domains and ranges very open and remove class specific cardinality and range restrictions (because there are no predefined classes to use in this way).

]

RDF/OWL/linked-data culture has a tendancy to seek widespread reuse of properties that have a common purpose or role for example a 'representativePoint' whereas the very nature of properties (attributes and association-roles) in UML is that their use is localised to a particular group of hierarchically organised classes. The property terms are less free (arguably not free at all) to be reused across application-schema.

In RDF the drive toward reuse is encouraged by open property domains, and for common purpose by more narrowly defined property ranges - though as general as is reasonably possible.

3.1.1 Foreign Properties and Property Reuse

There are two problems that we have not solved in any systematic way. The first, related to the discussion above is the problem of recognising properties that are general applicable across a wide range of UML class definitions (featuretypes and datatypes). It is highly likely that across the corpus of data specifications covering the 34 INSPIRE themes there are properties that are defined local to one class that are either suitable for use with another or more likely have in fact been re-invented in the context of another. These are properties that we would seek to give very liberal domain constraints to an then reuse rather than duplicate.

The second related problem that we have not tackled in any systematic way is how to incorporate the use of well known and widely used linked-data vocabularies such as e.g. FOAF [2] and ORG[3], or ISA core-vocabularies [4,5] where their capability of expression overlaps with that captured in an INSPIRE data-schema.

One approach is not to try, and to subsequently use rdfs:subClassOf and rdfs:subProperty of as a way to relate sufficiently compatible terms (generally casting the less widely know INSPIRE terms as subClass or subProperty of the more commonly known RDF/OWL terms).

Alternatively, one could undertake to select a number of such vocabularies to adopt and develop explicit mappings from INSPIRE attributes/association-roles to the relevant 'foreign' RDF properties and mapping from particular INSPIRE featureType classes to corresponding 'foreign' RDF classes (possibly materialised as an rdfs:subClassOf relation as above).

How important is it to adopt/reuse vocabularies from the wider linked-data community?

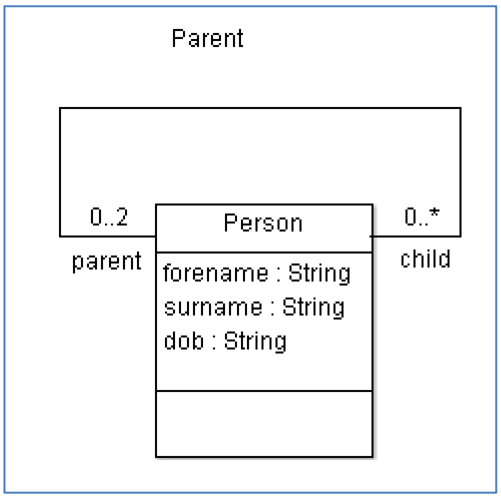
This is something of a two-way street. If there is an objective that the data be useful to a given community then it is most useful if presented to community in forms that they will recognise. On the other hand, if the data is valuable enough to a community they will adapt however it is presented.

The great power of linked data comes when links can be made between data published independently and this largely comes about through the use of shared vocabularies and identifiers.

Because our schema transformation activity to-date has been manual we have exercised judgement in these matters as we have been developing the RDF/OWL vocabularies.

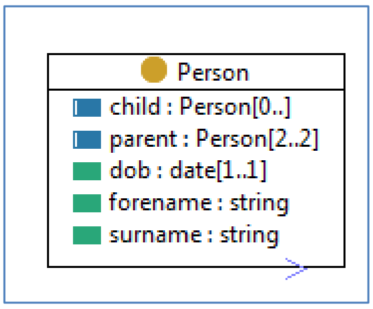
3.2 'What's the Subject'

Whilst its is evident from the previous section there are significant differences between RDF and UML in the scoping of properties, perhaps the more striking difference is in what's being said by the instantiation of an object and what it is that we are trying to constrain through the imposition of a schema or the application of an ontology.



Simplistically instance of UML objects can be thought of as boxes with labelled slots that represent attributes (roughly literal values[2]) or association-roles. They might be thought of as series of property/value pairs where some of the values are references to other objects. What a UML application-schema or model seeks to constrain is the set of properties (and possibly values) that can be used in the expression of a particular (UML) object instance. Roughly speaking, the UML instance is an information object (a record possibly) that models (or abstracts) some phenomenon in the world. A UML application-schema places constraints on what can be written in that record. The constraints are of course intended to reflect the constraints in the real-world. That a real-world person has two parents (also real-world people) can be modelled in an application-schema by having a Class of *Person* with an association *Parent* that has *parent* and *child* roles and a cardinality constraint on the *parent* role of at most 2.

With this model a valid *Person* object has *forename*, *surname*, *dob*, at most 2 *parent* and zero more *child* association(roles) and only that. What is being constrained is the record that can be kept about a person. The at most 2 constraint on the *parent* role avoids a potentially infinite expansion (because each parent is a person with 2 parents) - but that is a pragmatic constraint for the record/object, because the real-world person modelled by the instance does indeed have exactly two parents whether expressed in the data or not.



By contrast an RDF/OWL expression of a seemingly similar class says that members of the class *Person* are things that have exactly two *parents* that are also members of the class *Person*; that have zero or more children that are also members of the class *Person*; that have exactly one date of birth; and that have a *forename* and a *surname*[3]

The point being made here is that RDF/OWL modelling seeks express constraints on the 'thing' being abstracted - that a person (a real person) has two parents, whilst a UML/Object model seeks to express constraints on an instance of the model - that a Person object may reference up to two parent Person objects. In the later a Person objects models or abstracts a real-person, but it is not itself a real-person.

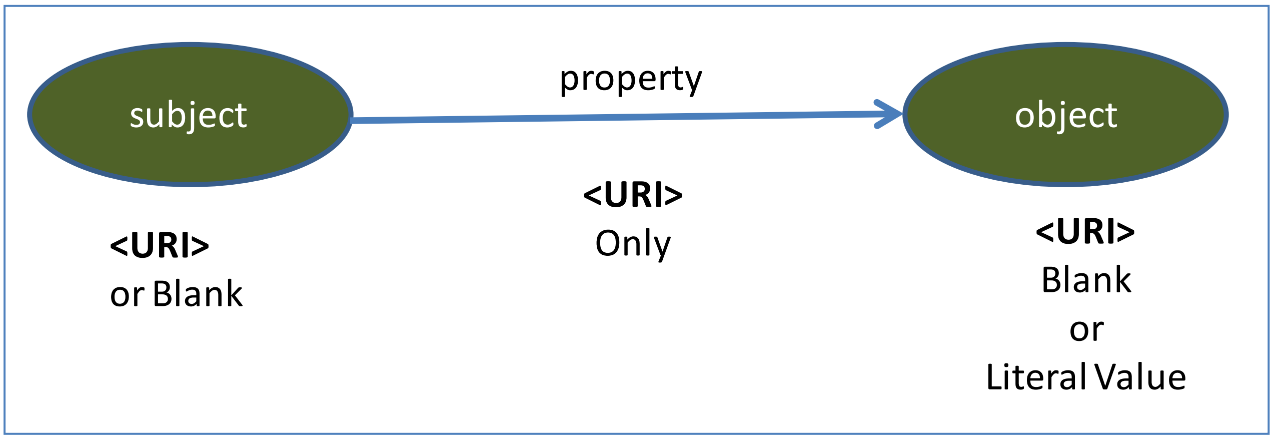
This seemly subtle distinction lies at the heart of some of the problems that arise when attempt to transform UML based INSPIRE application-schema into RDF/OWL vocabularies or ontologies.

One might view the *dob* field in an instance of our UML *Person* class as making a statement about the date of birth of a particular person. But there is no explicit reference to the intend real-world person. We might add a social-security number (assuming some world wide scheme for SSNs) as a thematic means to associate a Person-object with the real-world person assigned that SSN. However, the subject of any statement made by an object instance is somewhat implied and the tendancy in object

[2] which may be complex structured literals
[3] this is a somewhat naieve model of personal names. It should also be noted that in the absence of a cardinality restriction the OWL default is zero or more.

based systems is to identify the object rather than the thing that the object models or abstracts. Within the system the object stands 'proxy' for the thing that it models.

RDF on the other hand is made up of a collection of three part statements that at least make claims if not state facts[4]. Each statement has a *subject*, a *predicate* or *property* [5]and an *object*, often dubbed (s,p,o). Subjects are designated by URI (or blank) nodes; predicates are designated by URI only; and objects are designated by URI, blank nodes or literal values.



RDF graphs are made up of a collection of statements and each statement has an explicit subject node (even blank nodes). Admittedly the connection between an RDF node and some real-world phenomenon is no less illusory than the 'proxy' relation between a UML object instance and the real-world thing that it abstracts. The principle difference here is that in RDF one uses a subject identifier explicitly intended to designate the real-world thing about which statements are being made. In an object world, object identifiers to just that they identify the abstraction rather than the abstracted thing.

In RDF we might write:

```
station:MANPIC        a                     ex:RailywayStation ;
                      rdfs:label            "Manchester Picadilly Railway Station"@en ;
                      ex:numPlatforms       14 ;
                      ex:stationCode        "MANPIC"  ;
                      ex:representativePoint
                                    [ a            geo:Point ;
                                      geo:lat      53.477;
                                      geo:long ]   -2.230;
                      .
```

whereas as an INSPIRE railway station node would look something like:

nwrkRail:12345 ::RailwayStationNode	
name	Manchester Picadilly Railway Station
numPlatforms	14
stationCode	MANPIC
representativePoint	GM_POINT(-2.230,53.477)

The most significant difference between these two expressions is in the subject identifier in the RDF and the object identifier in the UML. The RDF subject identifier intentionally designates a particular railway station, while the UML object identifier designates an information object that abstracts/models a particular aspect of a given railway station (it's node like nature in a transport network).

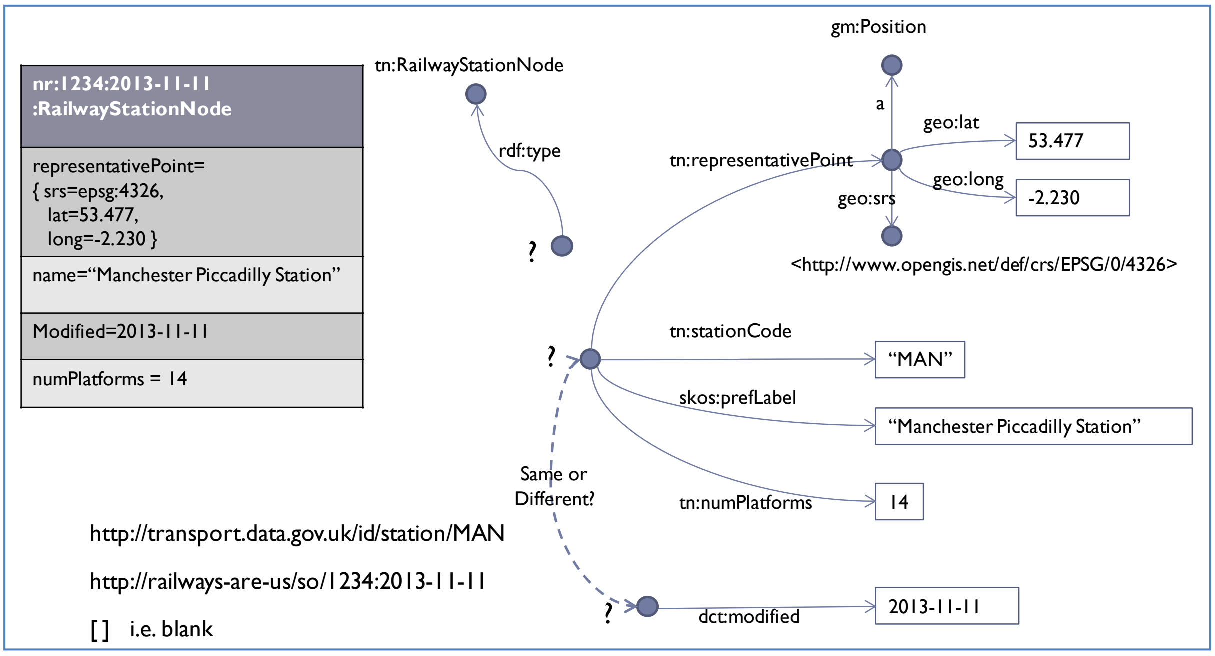
[4] In RDF is possible to write untruths just as in any language. There is no absolute truth/fact. When someone publishes an RDF document it contains statements which one assumes in good faith the publisher intends as true. In that sense one might regard them as claims made by the publisher.
[5] Predicate or property... predicates in logic are boolean value functions that evaluate true or false. RDF triples are sometime written in the form p(s,o) emphasing the nature of 'p' as binary predicate, with an extension, the set of (p,o) values for which 'p' evaluates 'true'. Property is more evoked as a the label on an 'arc' that connects subject to object is a statement.

In RDF multiple publishers can publish different accounts of the **same** station, and in a ideal world would use the **same** subject identifier to designate the station. In practice, publishers tend to mint their own URI for things and will typically use *owl:sameAs* statements to claim equivalence with the subject of some other account. This is most effective where there is a notion of reference URI sets from an authoritative source such that there are a small number of dominant reference URI that can be the target of these *owl:sameAs* expressions.

In INSPIRE/UML there can be multiple featureTypes that provide abstractions of the 'same' thing, whether from the same or different publishers. For example, there may be *RailwayStationNode* and *RailwayStationArea* spatial-objects (UML object instances) that abstract different aspects of, say, Manchester Piccadilly Railway Station. It is clear that these are necessarily different objects and therefore necessarily have different object identifiers. This again emphasises the distinction between subject identifiers (RDF) and object identifiers (INSPIRE/UML). In INSPIRE (and potentially RDF) *stationCode* serves as a thematic code which (loosely) indicates that there is some common 'thing' that the two objects are in some sense 'about' (models-of, abstractions-of).

3.2.1 Objects as Graphs

So... what in the RDF world is akin to a UML object instance?

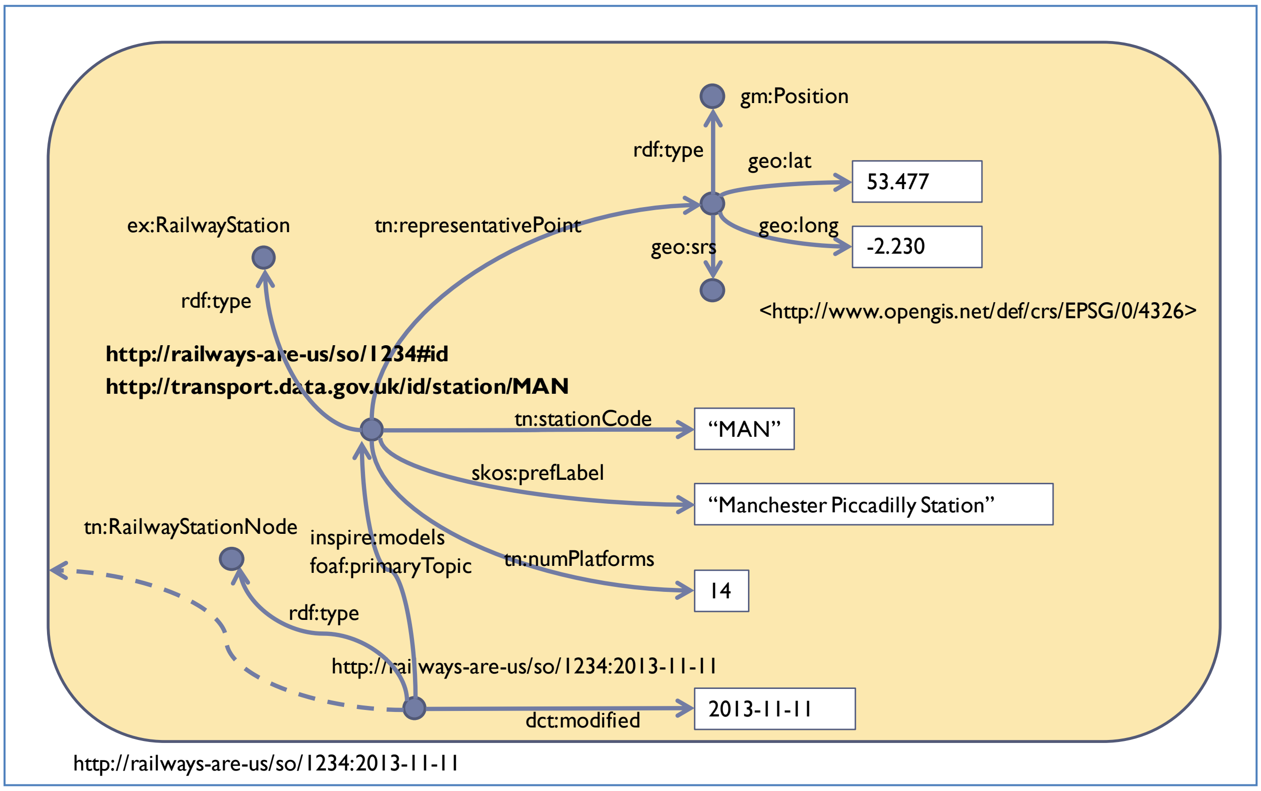


One possible view is to regard a small collection of statements about some subject as serving the role of an object in UML. Using the Trig[6] notation to extend our earlier example so that we can express RDF graphs with names:

```
nwrkRail:12345 {
  station:MAN
               a                      ex:RailwayStation ;
               rdfs:label             "Manchester Picadilly Railway Station"@en ;
               ex:numPlatforms        14 ;
               ex:stationCode         "MAN" ;
               ex:representativePoint
                              [ a            geo:Point ;
                                geo:lat      53.477;
                                geo:long ]   -2.230;
               .
  nwrkRail:12345
               a                      inspire-tn:RailwayStationNode ,
                                      foaf:Document ;
               dct:modified           "2013-11-11"^^xsd:date ;
               dct:creator            [ a foaf:Person ; foaf:name "Stuart Williams"@en ] ;
               dct:publisher          <railway-stations-are-us-publishing> ;
               foaf:primaryTopic      station:MANPIC ;
```

}

Or pictorially


gm:Position
rdf:type
geo:lat
53.477
geo:long
-2.230
geo:srs
<http://www.opengis.net/def/crs/EPSG/0/4326>
ex:RailwayStation
tn:representativePoint
rdf:type
http://railways-are-us/so/1234#id
http://transport.data.gov.uk/id/station/MAN
tn:stationCode
"MAN"
skos:prefLabel
"Manchester Piccadilly Station"
tn:RailwayStationNode
inspire:models
foaf:primaryTopic
tn:numPlatforms
14
rdf:type
http://railways-are-us/so/1234:2013-11-11
2013-11-11
dct:modified
http://railways-are-us/so/1234:2013-11-11

Notice that within the object-graph there are two statement subjects - the real-world station and the object itself. This allows statements to be made separately about the station and the object. Once can clearly express provenance of the object (object metadata) distinct from data expressed about the station.

However, what of the constraints that an INSPIRE data specification application schema might impose?

In this formulation INSPIRE application schema would constrain the properties that could be used in speaking **about** a railway station within the context of say a *RailwayStationNode* to just those that allowed of such an object. For example, the object above would be malformed, as an INSPIRE *RailwayStationNode* if it made statements about Manchester Piccadilly that need to be made in the context of a *RailwayStationArea* object e.g. the geometric extend of the railway station.

Unfortunately, constraints on the form of what can be said are not what RDFS and OWL do. They establish what can be inferred from the statements that are made and when a collection of statements is inconsistent in the sense that they could not all be simultaneously true about the entities described by the statements.

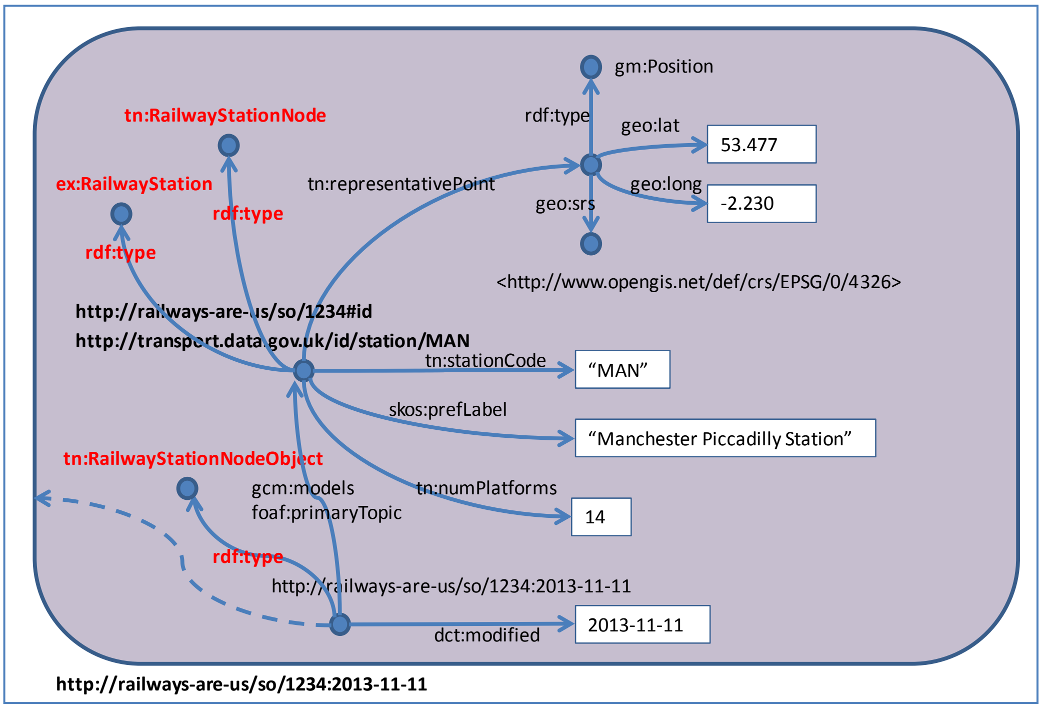
[Post-Workshop meeting on 15th April]

One other question that surfaces is what *rdf:type* statements should be applied to which graph nodes particularly the nodes that represent the spatial-object (as a graph) and the 'thing' that the spatial-object is an abstraction of, in this case a railway station.

One option is to leave property domains and ranges very wide open, in which case spatial-object type naturally accrues to the graph node that represents the spatial-object (as a graph) as in the diagram above. However, if there is a requirement to be more constrained about the domains and ranges of properties that pertain to the abstracted 'thing', ie. the railway station, then this is probably best served by creating two derived RDF/OWL classes - one that can act as a 'mix-in' type for typing the abstracted thing (a railway station as a tn:RailwayStationNode) which can be used to specify property domains and ranges where necessary, and a second RDF/OWL class to type the spatial-object - say tn:RailwayStationNodeObject so that objects of a given type can be found. These two changes as illustrated in red in the revised TRIG fragment below.

```
nwrkRail:12345 {
  station:MAN
        a                       ex:RailwayStation ,
                                tn:RailwayStationNode ; ## Derived 'mix-in' class
        rdfs:label              "Manchester Picadilly Railway Station"@en ;
        ex:numPlatforms         14 ;
        ex:stationCode          "MAN" ;
        ex:representativePoint
                       [ a             geo:Point ;
                         geo:lat       53.477;
                         geo:long ]    -2.230;
        .
  nwrkRail:12345
        a                       inspire-tn:RailwayStationNodeObject ,
                                foaf:Document ;
        dct:modified            "2013-11-11"^^xsd:date ;
        dct:creator             [ a foaf:Person ; foaf:name "Stuart Williams"@en ] ;
        dct:publisher           <railway-stations-are-us-publishing> ;
        foaf:primaryTopic       station:MAN ;
        .
}
```

Or pictorially


gm:Position
tn:RailwayStationNode
rdf:type
geo:lat
53.477
ex:RailwayStation
tn:representativePoint
geo:long
-2.230
geo:srs
rdf:type
rdf:type
<http://www.opengis.net/def/crs/EPSG/0/4326>
http://railways-are-us/so/1234#id
http://transport.data.gov.uk/id/station/MAN
tn:stationCode
"MAN"
skos:prefLabel
"Manchester Piccadilly Station"
tn:RailwayStationNodeObject
gcm:models
foaf:primaryTopic
tn:numPlatforms
14
rdf:type
http://railways-are-us/so/1234:2013-11-11
2013-11-11
dct:modified
http://railways-are-us/so/1234:2013-11-11

The group consensus at the workshop meeting was largely to leave property domains and ranges very open, possibly as wide as *rdfs:Resource* in all cases.

[End of post meeting insert]

While the object-graph approach to framing INSPIRE spatial object illustrated above is appealing since it clearly separates object identity from subject identity, it also introduces problems of its own.

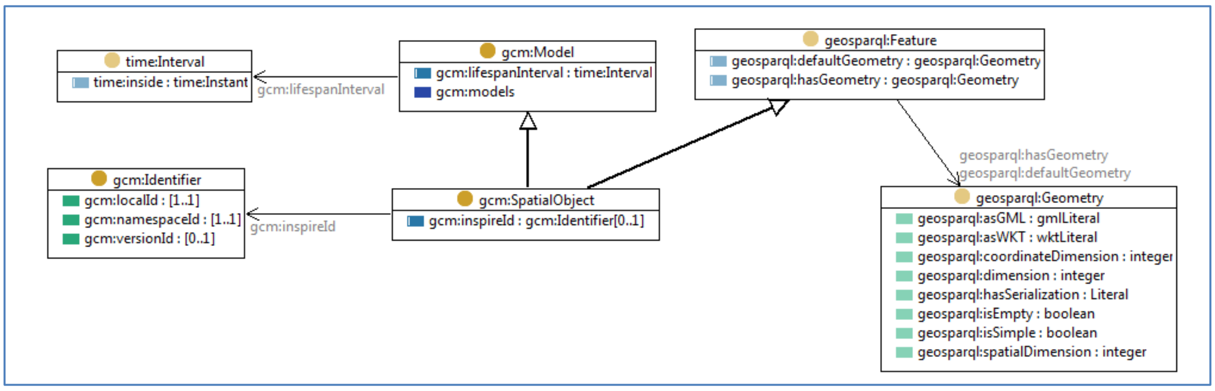
Firstly, as mentioned above, there is the problem that neither RDFS or OWL are capable of expressing syntactic constraints on an RDF graph.

Secondly, there are difficulties if a collection (e.g. a dataset) of object-graphs formed this way are 'poured' into a triplestore or indeed a quadstore. Within a triplestore all the object boundaries are lost and in general it would be impossible to segregate triples back into their original object graphs. In a quadstore object-graph separation can be maintained, and the common practice of a using a UNION default graph provides and environment for querying across multiple objects/graphs. However, SPARQL the queries become much more complicated - if for example one wants to use object metadata to qualify the objects of interest in a query.

In our work to date, despite the 'purity' of the object-graphs approach from an RDF point-of-view which allows statement subjects to designate real-world entities being spoken of, we have taken a different, more pragmatic general approach that can operate within a single graph triplestore.

3.2.2 Spatial-Objects as Nodes

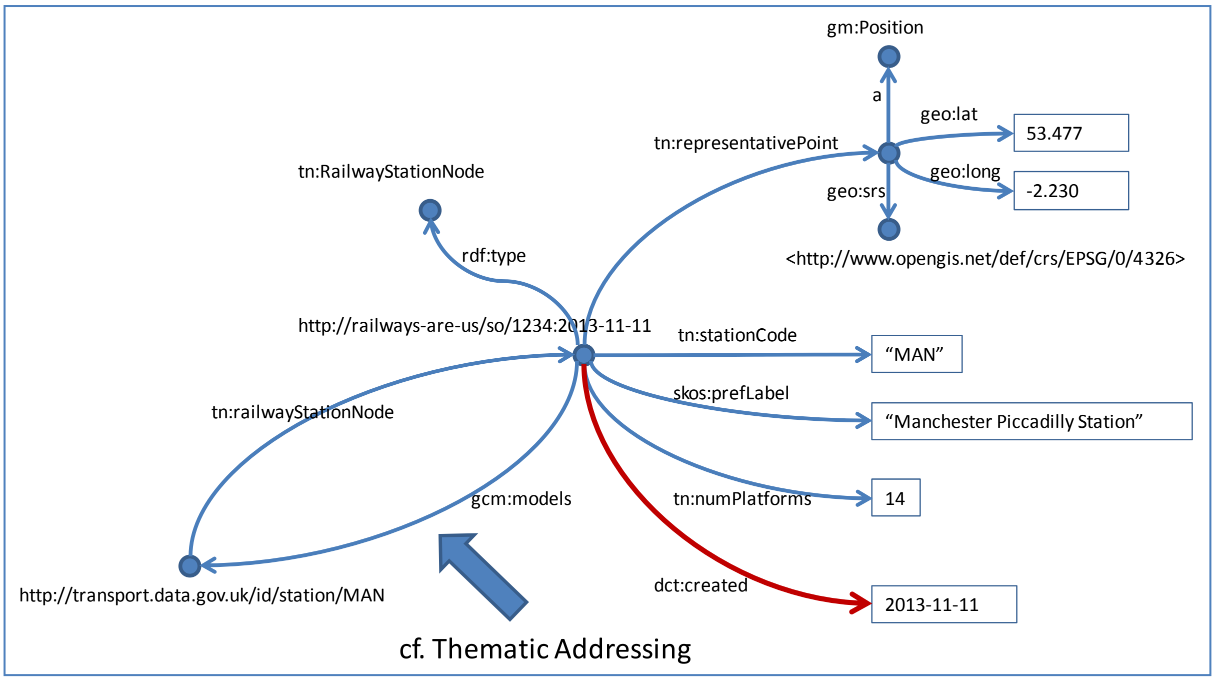
The approach we have adopted is to formulate objects, spatial-objects, as nodes in an RDF graph. This allows much more use of RDFS and OWL in describing RDF models/vocabularies derived from INSPIRE application schema. However, it does expose the potential to mistakenly conflate spatial-objects with the things which they model/abstract.



time:Interval
time:inside : time:Instant
gcm:lifespanInterval
gcm:Model
gcm:lifespanInterval : time:Interval
gcm:models
geosparql:Feature
geosparql:defaultGeometry : geosparql:Geometry
geosparql:hasGeometry : geosparql:Geometry
geosparql:hasGeometry
geosparql:defaultGeometry
gcm:Identifier
gcm:localId : [1..1]
gcm:namespaceId : [1..1]
gcm:versionId : [0..1]
gcm:inspireId
gcm:SpatialObject
gcm:inspireId : gcm:Identifier[0..1]
geosparql:Geometry
geosparql:asGML : gmlLiteral
geosparql:asWKT : wktLiteral
geosparql:coordinateDimension : integer
geosparql:dimension : integer
geosparql:hasSerialization : Literal
geosparql:isEmpty : boolean
geosparql:isSimple : boolean
geosparql:spatialDimension : integer

We define a small vocabulary to capture some notions from the INSIRE Generic Conceptual Model. We present spatial-object as a subclass of a more general notion of a model and provide a property, *gcm:models*, whose role is to link a spatial-object to the real-world thing that it abstracts. The use of *gcm:models* of is optional. There is often no handy, widely used/adopted, reference URI for the entity that a spatial object models. In many respects *gcm:models* serves the role of thematic referencing in INSPIRE. It provides a way to explicitly make the link between spatial-object and the thing that it abstracts. It can serve as a 'join' point between multiple spatial-objects that abstract (aspects of) the same thing, whether multiple spatial object published by the same publisher or spatial-objects from different publishers.

```
nwrkRail:12345
       a                     tn:RailwayStationNode ,
                             gcm:SpatialObject ;
       gcm:models            stations:MAN ;
       gcm:inspireId  [
                     gcm:nameSpaceId       netwrkRail: ;
                     gcm:localId           "12345" ;
              ] ;
       rdfs:label            "Manchester Picadilly Railway Station"@en ;
       ex:numPlatforms       14 ;
       ex:stationCode        "MAN" ;
       ex:representativePoint
                     [ a           geo:Point ;
                       geo:lat     53.477;
                       geo:long ]  -2.230;
       dct:created    "2013-03-30"^^xsd:date ;
       dct:creator    [ a foaf:Person ; foaf:name "Stuart Williams"@en ] ;
       dct:publisher  <railway-stations-are-us-publishing> ;
       .
```



We include an explicit *gcm:inspireId* property to carry the object's INSPIRE spatial-object identifier, however, this field is potentially redundant in the face of a URI serving as persistent object identifiers. For compactness here, we have shown the value of *gcm:inspireId* as blank-node however, as there some pragmatic issues with blank nodes we would typically use a URI derived from the spatial-object's URI. e.g. in this case:

```
nwrkRail:12345/inspireId
```

Note that the subject URI used in making statements about a real-world thing is the spatial-object URI rather than the real-world subject URI. This risks conflation of object with real-world thing.

One way to square this is to regard spatial-object instances as being like features on a map, i.e. features are abstract representations of real-world phenomenon that inhabit an abstracted world (a map) and occupy position and/or space within that abstracted world - after all the spatial-reference systems with which we express position and geometry are themselves abstractions.

There are issues here. We have two entities with distinct URI, but in writing RDF we are infact using the spatial-object URI as a subject designator to make statements about the real-world thing. How can we now speak clearly about the object itself without mis-speaking about the real-world thing? For example the 'creation' date of Manchester Piccadilly Railway Station might be taken is at the date when it first opened in 1842[6] the 'creation' date of the corresponding spatial object, an information abstraction, is somewhat more recent. Also the creators of these two things are different and it only makes sense to speak of the spatial-object as having a publisher.

Using spatial-objects as subject nodes make speaking clearly and distinctly about spatial-object and the thing it abstracts/models somewhat more difficult.

The W3C TAG has worked [7] on this topic and suggest an approach for annotating property definitions to indicate whether they are used to make direct assertions about the immediate statement subject or whether they are used indirectly to make assertions about something else. However, at the time of writing we don't believe there is a concrete annotation vocabulary in common use to support this practice.

[6] http://en.wikipedia.org/wiki/Manchester_Piccadilly_station

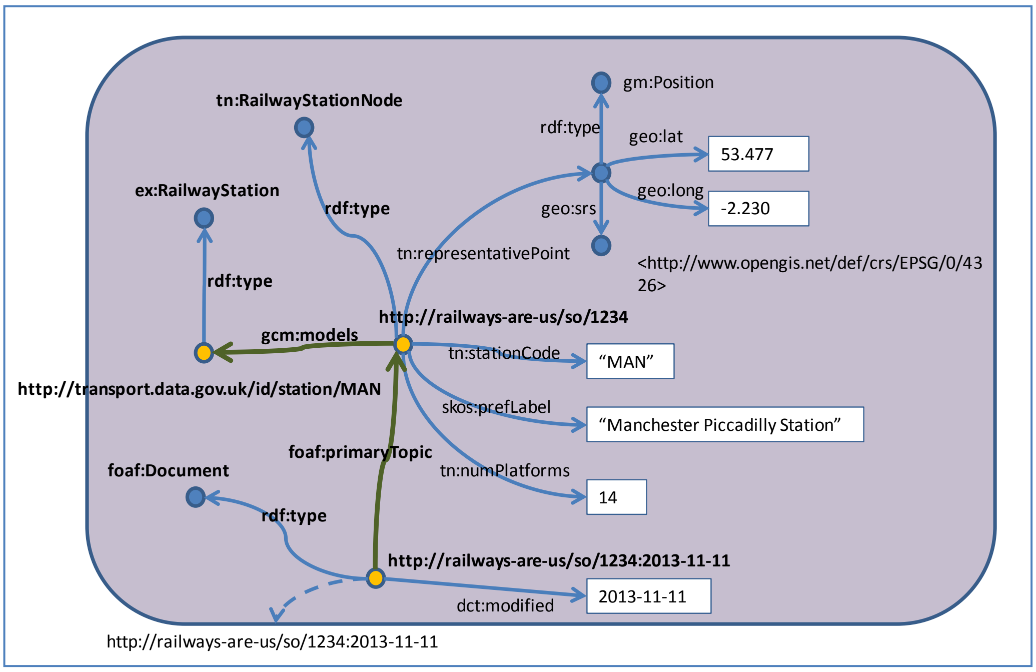
3.2.3 Spatial-Objects as Graphs and Nodes

Just to draw the previous two subsections together we can merge ideas from both, which potentially address the conflation issue mentioned above. Here we have

- A real-world station, `stations:MAN` modelled by
- a spatial-object - `nwrkRail:12345` described in a
- a graph/document `nwrkRail:12345.doc`

Here we regard a spatial-object as an information abstraction that models a real-world phenomenon and which is itself described in a document. In this way, publication metadata associated with the spatial-object publication can be associated with the document that describes it - particularly if the document is a somewhat ephemeral artifact produced, say by a query engine, for the purposes of transferring the state of the object - ie. the object really is the signifcant published artefact rather than the document.

```
nwrkRail:12345.doc {
nwrkRail:12345
        a                       inspire-fn:RailwayStationNode ,
                                gcm:SpatialObject ;
        foaf:isPrimaryTopicOf   nwrkRail:12345.doc
        gcm:models              stations:MAN ;
        gcm:inspireId   [
                gcm:nameSpaceId         netwrkRail: ;
                gcm:localId             "12345" ;
        ] ;
        rdfs:label              "Manchester Picadilly Railway Station"@en ;
        ex:numPlatforms         14 ;
        ex:stationCode          "MAN" ;
        ex:representativePoint
                        [ a             geo:Point ;
                          geo:lat       53.477;
                          geo:long ]    -2.230;
        .
nwrkRail:12345.doc
        a                       foaf:Document ;
        foaf:primaryTopic       nwrkRail:12345 ;
        dct:created             "2013-11-11"^^xsd:date ;
        dct:creator             [ a foaf:Person ; foaf:name "Stuart Williams"@en ] ;
        dct:publisher           <railway-stations-are-us-publishing> ;
                .
}
```



The spatial-object `nwrkRail:12345` is now clearly articulated; we make a pragmatic, rather than purist, choice to use the spatial-object URI as a subject for making statements whose real subject is the modelled real-world 'thing'; and we use a document URI, `nwrkRail:12345.doc`, as a subject URI to make statements that are really about the spatial-object (spatial-object metadata). There is also a residual tension about whether the graph URI should follow the spatial-object URI or the document URI.

3.2.4 Spatial Objects as... Summary

To summarise the preceding subsections:

Spatial-objects a graphs

The natural linked-data style is to speak directly a real-world 'thing' or phenomenon using RDF statements that use a subject URI which designates said real-world 'thing'. INSPIRE application-schema seek place constraints on the range of statements that can be made together about a given subject - in the sense that they seek to ensure some given aspect (and maybe only that aspect) of the real-world thing is addressed by given object - e.g. the aspects of a railway station required to present information about it in the form of a network model. In this formulation, spatial-objects are naturally graphs with constraints on the properties that can be used in the graphs. With such direct reference to the real-world subject neither RDFS nor OWL can be used to express the required constraints.

One possibility is to take a view that a real-world thing is simultaneously, say, an *ex:RailwayStation*, *inspire-tn:RailwayStationNode* and *inspire-tn:RailwayStationArea*. However, we suspect such mixin usage to be somewhat counter to INSPIRE culture.

Spatial-Objects as nodes

Instead of grouping the RDF statements of a given spatial-object by graph containment, we can group them by RDF subject node where the subject URI designates a given spatial-object. However, the statements that comprise the spatial-object, indeed the spatial-object itself stand proxy for the real-world 'thing' modelled by the spatial-object. Because RDF statements made using the spatial-object URI as a subject, it is now difficult to make statements about the spatial-object itself (spatial-object metadata) without conflating the 'thing' modelled by the spatial object with the spatial-object. Consider

```
nwrkRail:12345         dct:created           "2014-04-01"^^xsd:date ;
                       inspire-tn:numPlatforms       14
                       .
```

Manchester Piccadilly Railway Station does indeed have 14 platforms, whilst as an information object, `nwrkRail:12345` has none. On the other hand, `nwrkRail:12345` was created relatively recently (for the purposes of this document) whereas Manchester Piccadilly Railway Station has existed in one form or another for several tens of years. However, the constraints expressed by INSPIRE application-schema can be applied (at least to some degree) using RDFS/OWL because it is the description of the spatial-object that is constrained, not what can be said in general about the modelled real-world 'thing'.

Spatial-objects as Nodes and Graphs

Retains most of the characteristics of the spatial-objects as Nodes formulation, but adds a document node which is used as a graph name (positioning the graph itself as a document that describes both itself and a spatial-object or objects). There remains a to some extend unwelcome degree of indirection in that the statements made using the spatial-object URI as a subject are infact about the modelled 'thing' and statements made using the document URI are (arguably) infact about the spatial-object. The latter is less of an issue because with a 1-1 correspondence between document and spatial-objects their conflation is much less problematic that conflation of spatial-object and modelled 'thing'.

Our deployments to date have largely followed this last pattern, although the object metadata has largely been ephemeral, generated in responding to a request rather than having been materialised literally in the stored data. Partially this has been due to a lack of data about the object or database records we've been handling - so we have just not had the object metadata to work with in the first place.

3.3 Versioned Spatial-Objects

INSPIRE spatial-object identifiers are composed of three elements: a *namespaceId* assigned by member-states to spatial object publishers; a *localId* to differentiate between spatial-objects within the same namespace; and an optional *versionId* to different, presumably, between different versions of the 'same' spatial object.

INSPIRE is not overly prescriptive on the use of spatial-object versions. In world where whole collections of objects are published as coherent data set releases this is not much of a problem. VersionIds may reflect the versionId of the dataset as a whole. Even if objects are independently versioned, a given data set release can still be published with a single-current version for each included spatial object.

Publishing spatial-objects as linked data may involve successive 'whole dataset' publications or it may be more incremental where updates are made at arbitrary times to individual spatial objects. We see three possible versioning models:

Single current version:

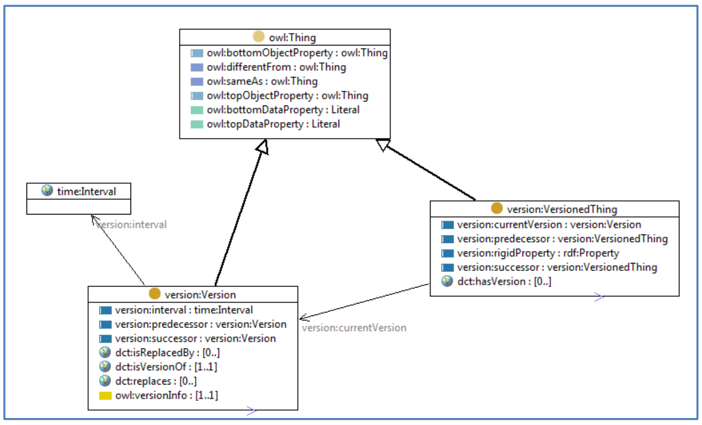
A publisher only ever publishes a single version of a spatial object at a given time. The object is update in place and bears no explicit versionId. namespaceId and localId alone are sufficient to identify the object. Object metadata (if any - see comments about conflation of object and modelled entity) such as modification date/time might give hints that the object has changed.

Single current version with *versionId*

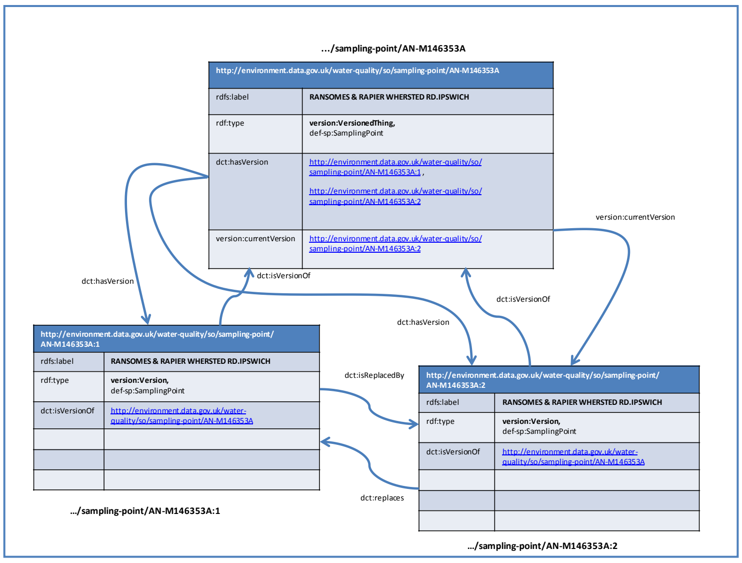
As above only a single version of the object is published and maintained by the publisher at any given time. However, the object carries and explicit *versionId* as an attribute as a means to signal change. The *versionId* does not contribute to the object's identity, which only requires *namespaceId* and *localId*. Version history is not accessible.

Enduring spatial-objects with versioned temporal parts (versions).

A spatial-object is formulated as an enduring spatial-object which serves to collect versioned snapshots (a temporal-part [8] or *Version*) that represent the state of the object at different points during it's lifetime. The enduring object (a *VersionedThing*) represents the object over its entire lifetime and even beyond. In our deployments we also maintain a non-monotonic "currentVersion" link as shortcut link to the most recent 'snapshot' (a Version). *VersionedThing* and *Version* act as 'mixin' types added to spatial object nodes. VersionedThings are somewhat vestigial and should only really carry 'rigid' properties, properties that are invariant across spatial-object versions. The diagram below illustrates the small versioning vocabulary[7] that we have created and its use in a project to publish water quality information as linked data.


owl:Thing
owl:bottomObjectProperty : owl:Thing
owl:differentFrom : owl:Thing
owl:sameAs : owl:Thing
owl:topObjectProperty : owl:Thing
owl:bottomDataProperty : Literal
owl:topDataProperty : Literal
time:Interval
version:interval
version:VersionedThing
version:currentVersion : version:Version
version:predecessor : version:VersionedThing
version:rigidProperty : rdf:Property
version:successor : version:VersionedThing
dct:hasVersion : [0..]
version:Version
version:interval : time:Interval
version:predecessor : version:Version
version:successor : version:Version
dct:isReplacedBy : [0..]
dct:isVersionOf : [1..1]
dct:replaces : [0..]
owl:versionInfo : [1..1]
version:currentVersion

[7] http://purl.org/linked-data/version


.../sampling-point/AN-M146353A
http://environment.data.gov.uk/water-quality/so/sampling-point/AN-M146353A
rdfs:label
RANSOMES & RAPIER WHERSTED RD.IPSWICH
rdf:type
version:VersionedThing, def-sp:SamplingPoint
dct:hasVersion
http://environment.data.gov.uk/water-quality/so/sampling-point/AN-M146353A:1 ,
http://environment.data.gov.uk/water-quality/so/sampling-point/AN-M146353A:2
version:currentVersion
http://environment.data.gov.uk/water-quality/so/sampling-point/AN-M146353A:2
version:currentVersion
dct:hasVersion
dct:isVersionOf
dct:isVersionOf
dct:hasVersion
http://environment.data.gov.uk/water-quality/so/sampling-point/AN-M146353A:1
rdfs:label
RANSOMES & RAPIER WHERSTED RD.IPSWICH
rdf:type
version:Version, def-sp:SamplingPoint
dct:isVersionOf
http://environment.data.gov.uk/water-quality/so/sampling-point/AN-M146353A
dct:isReplacedBy
dct:replaces
.../sampling-point/AN-M146353A:1
http://environment.data.gov.uk/water-quality/so/sampling-point/AN-M146353A:2
rdfs:label
RANSOMES & RAPIER WHERSTED RD.IPSWICH
rdf:type
version:Version, def-sp:SamplingPoint
dct:isVersionOf
http://environment.data.gov.uk/water-quality/so/sampling-point/AN-M146353A
.../sampling-point/AN-M146353A:2

On the web, one of the 'mantra' is "Cool URIs never die..." Creating URI that of the form:

```
    {namespaceId}/{localId}[:{versionId}]
```

Means that *versionId* becomes part of a potentially bookmark-able, reference-able, cite-able web identifier and may be required to persist for a long time.

Typically we will create a series of related access URI as illustrated below. Our specific practice is evolving, but is along the lines shown below.

URI Pattern	**Behaviour**
`{namespaceId}/{localId}`	References and accesses the enduring entity
`{namespaceId}/{localId}:{versionId}`	References and accesses a specific version/snapshot
`{namespaceId}/{localId}[:{versionId}]/current`	An API to retrieve the current version snapshot relative to a specific version or the enduring entity.
`{namespaceId}/{localId}[:{versionId}]/versionAt/{yyyymmdd[:hhmm]}`	An API to retrieve the version snapshot current at a particular date/time. (in the past)

`{namespaceId}/{localId}:{versionId}/previous`
`{namespaceId}/{localId}:{versionId}/next`
	An API to retrieve previous or next versions relative to some specific version.

The key point is that where {versionId} become part of URI that identify particular spatial-object versions there is as need to think about persistence of both the identifier and the data behind it. We take a largely monotonic approach, retaining version history.

When making reference to a spatial-object in from published data, we invariably refer to the enduring entity (a *version:VersionedThing)* rather than a specific version (*version:Version*). It most case it terms of publish data, it is almost impossible to anticipate the temporal context a given application would be in interested in. Instead, by reference to the enduring spatial-object - which remains reference-able (in principle) over all future time (much as we can refer to long-deceased playwrights well after their demise) and ideally accessible - a particular application can determine the temporal scope of interest and access the relevant temporal snapshots.

4 Persistent URI and INSPIRE

4.1 UK Generic Persistent URI Patterns

In the UK we have been developing patterns for the use of URI as persistent identifiers for all manner of things. The orginal work [URISets] was focussed on URI of the general form:

```
http://{sector}.data.gov.uk/...
```

where {sector} represents a thematic division of the URI space around themes such as 'health', 'education', 'environment', 'legislation', 'transport' and so forth. One of the practical difficulties that arises with this approach is that it inherently creates multi-tenanted divisions within URI space which are difficult to administer and govern. Within any given sector there are potentially multiple organisations or sub-organisations that have data to publish. At a practical level the (sub-)organisation publishing data needs to organise and administer infrastructure to support their data publication. Coordinated sharing of the URI space in any given sector is problematic.

Also, at least within the UK sector, the original guidance was framed very much in the context of data.gov.uk. Trading funds, devolved administrations, and local authorities all have legitimate reasons to want to publish data at URI outside of the orginal {sector}.data.gov.uk pattern.

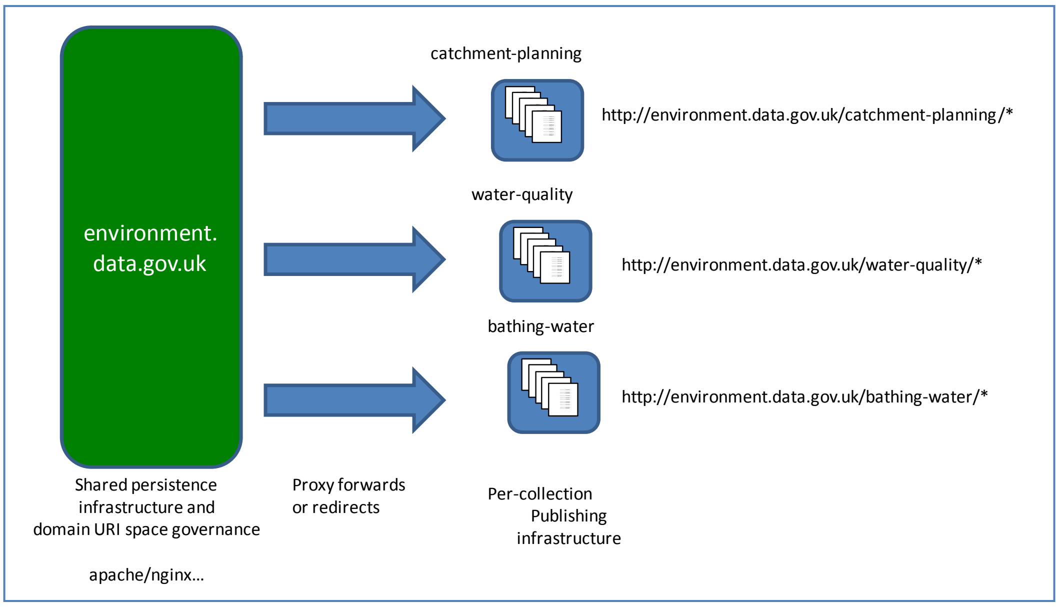
Revisions currently in progress propose a different to level pattern:

```
{prefix} = http://{domain}{/collection*}...
```

Here `{domain}` is basically any internet domain name. Of course the persistence of URI is rooted in persistence of domain name and that is vested ultimately in the organisational entities that own/adminsiter those domains.

`{/collection*}` is zero or more segments of URI path that distinguish between collections published under a domain. A collection is a cohesive group of vocabularies, reference data, datasets and data items that are administered as a unit. The governance notion is that were responsibility for a collection to move within or between organisations, the collection itself is regarded as an atomic unit which would not be 'broken-up' and dispersed to different (sub-)organisations.

In-order to provide persistence in the face of change the intention is that the requests are handled by two levels of infrastructure. The first - administered by the organisational owner of the `{domain}`, which could be a stakeholder group in the case of a shared domain, provides for persistent access URI to all colllections organised within that domain. Requests are routed to a second tier of infrastructure supporting a particular collection either by proxy or by redirection. Proxy forwarding is more transparent in the sense that it is less apparent to a requesting client, but it leaves the proxy in the data path for any responses; redirections are more evident to a requesting client, the request URI is also visibly rewritten through the redirection process however, the proxy is **not** involved in the response data path, which is direct between client and the infrastructure serving a collection.


catchment-planning
http://environment.data.gov.uk/catchment-planning/*
water-quality
environment.
data.gov.uk
http://environment.data.gov.uk/water-quality/*
bathing-water
http://environment.data.gov.uk/bathing-water/*
Shared persistence infrastructure and domain URI space governance
Proxy forwards or redirects
Per-collection Publishing infrastructure
apache/nginx…

This leaves us with a layer of persistent URI and a layer of infrastructure URI. The right-hand end patterns of both persistent and infrastructure URI are likely to mirror each other. Should it be necessary to move a collection onto different infrastructure with different infrastructure URI, the mapping between persistent URI and infrastructure URI maintained by the proxy layer can be updated to reflect the new infrastructure location of the collection. In this way access via persistent URI is maintained.

The draft revised UK guidance also acknowledges that an indefinite persistence commitment is impractical for both political and financial reasons. It advises the use of metadata to provide an expression of the commitment the publisher of a collection. At this time there is no concrete vocabulary proposed for the metadata, but it would be expected to indicate how far into the future the URI space will persist and how much notice would be given (through revised metadata) of any intention to retire-from or relocate-within. In the case of relocations within persistent URI space it is also good practice to install redirects from the old location the new which should also be maintained for some advertised interval.

The table below summarises the patterns proposed in the draft revised UK guidance[10] where **`{prefix}`** is as above:

	Pattern

URI Sets	`{prefix}`**`/id`**`/{concept} or`
	`{prefix}/{concept}`**`#id`**

Identifier URI
(for reference items)
	`{prefix}`**`/id`**`[/{concept}/{key}]* or`
	`{prefix}[/{concept}/{key}]*`**`#id`**

Document URI
(for reference data)
	`reference data for single reference items:`
	`{prefix}`**`/doc`**`[/{concept}/{key}]* or`
	`{prefix}[/{concept}/{key}]*`
	`optionally, reference data for lists of reference items`
	`{prefix}`**`/doc`**`/{concept}/{key}]*/{concept} or`
	`{prefix}[/{concept}/{key}]*/{concept}`

Vocabulary URI **(for vocabularies, ontologies, concept schemes, codelists and schema)**	`{prefix}`**`/def`**`{/vocabulary*}`

Vocabulary Term URI
(for term definitions within a vocabularies, ontologies, concept schemes, codelists and schema)
	`{prefix}`**`/def`**`{/vocabulary*}/{term} or`
	`{prefix}}`**`/def`**`{/vocabulary*}#{term}`

Dataset URI **(for datasets)**	`{prefix}`**`/data`**`{/dataset*}`
Data Item URI **(for data items within datasets).**	`{prefix}`**`/data`**`{/dataset*}[/{concept}/{key}]*`

4.2 URI for Location

The UK also developed a second tranche of guidance focus on the use of URI for the purpose of publishing INSPIRE spatial-objects as linked-data. The original guidance [9] focussed on:

a. URI for Spatial-Objects: by embedding the components of an INSPIRE spatial object identifier within an HTTP URI, thus:

```
http://location.data.gov.uk/so/{theme}/{featureType}/{namespaceId}/{localId}[:{versionId}]
```

Note that {theme} and {featureType} do not actually contribute to the identification of the object. Whilst they provide (humanly) useful hints about the nature of the object, they complicate both the URI space, particularly of a group of spatial-objects of mixed type; and they complicate the process of deriving a URI for a spatial-object - because more need to be known about the spatial object than just the components of its identifier.

b. URI for derived vocabulary terms of the form:

```
http://location.data.gov.uk/def/{theme}[/{package}][/({concept}|{class})][/{version}]/{term}
```

c. URI for real-world things modelled by spatial-objects:

```
http://{sector}.data.gov.uk/id/{concept}/{reference}
http://location.data.gov.uk/id/{theme}/{concept}[/{codeset}]/{reference}
```

The first pattern is for the case where some authoritative source has already assigned an authoritative reference identifier. The second pattern is so that a URI can assigned within the location.data.gov.uk space based on a thematic code.

At the time this guidance was written the syntactic rules were such that it was not possible to use a URI (of any kind) as an INSPIRE namespace. However, in the period since that guidance was published there has been a change which allows URI in general and HTTP URI in particular to be used as INSPIRE namespaces. This change allows for the persistent URI for spatial-objects to now be based more naturally within the collection(s) which publish them.

Based on the revised generic URI patterns, we have reformulated the URI for spatial-objects [11] as follows.

From the generic pattern:

```
{prefix} = http://{domain}{/collection*}
```

an INSPIRE namespace identifier is formulated as:

```
{inspireNamespaceUri} = {prefix}[/so][/{class}]
```

resulting in URI for INSPIRE spatial objects of the form:

```
{inspireNamespaceUri}/{localId}[:{versionId}] or
{inspireNamespaceUri}/{localId}[/{versionId}] or
{inspireNamespaceUri}/{localId}[/version/{versionId}]
```

Note the retained, but optional use of `{class}` as part of the `{inspireNamespaceId}`. It is retained for the benefit of those who found some previous utility in its presence. Also type marker '/so' is optionally retained. Many find it a useful hint indicating that the referrent is as spatial-object of some kind. However, that may already be evident from the lower order parts of a collection name. We have left somewhat open how to combine {localId} and {versionId} into a URI. The differences a largely in the spelling of a separator as ':' or '/' or '/version/'.

`{localId}[:{versionId}]`
: combines these elements within a single URI path segment. With some technology bases it is more convenient to be able to handle localId and versionId as a single unit, breaking it into its component parts when necessary.

`{localId}[/{versionId}]`
: makes the elements independent URI path segments. This works better for technologies bases that have some difficulty in micro-parsing a URI path segment.

`{localId}[/version/{versionId}]`
: fits with the more verbose pattern of ({param}/{value})* for access to subordinate structure or multi-dimensional observation datasets.

Ideally we will resolve these patterns down to a single pattern. The authors current preference is the first, ie. `{localId}[:{versionId}]`. The third pattern is verbose, even if the separator were reduced to just '`/v/`'.

The following URI pattern were proposed in a recent Environment Agency project to develop models to support the publication of catchment-planning information as linked-data in response to the requirements of the EU Water Framwork Directive. Later sections will discuss more of the modelling the modelling work undertaken by that project.

Entity Type	**Proposed Instance URI pattern**
`RiverBasinDistrict`	`http://environment.data.gov.uk/catchment-planning /so/am/RiverBasinDistrict/{riverBasinDistrictKey}[:{version}]`
`Catchment`	`http://environment.data.gov.uk/catchment-planning /so/am/Catchment/{catchmentKey}[:{version}]`
`WaterBody`	`http://environment.data.gov.uk/catchment-planning /so/am/WaterBody/{waterBodyKey}[:{version}]`
`WaterbodyCatchment`	`http://environment.data.gov.uk/catchment-planning /so/am/WaterBodyCatchment/{waterbodyKey}[:{version}]`
`ProtectedArea`	`http://environment.data.gov.uk/catchment-planning /so/am/ProtectedArea/{protectedAreaKey}[:{version}]`

AdminstrativeArea	http://environment.data.gov.uk/catchment-planning /so/am/AdminsitrativeArea/{adminAreaKey}[:{version}]

We allowed ourselves the slight liberty of retaining a {theme} path segment, 'am' in order to better align with earlier pilot work. Note the use of 'catchment-planning' as collection name. As far as possible collection names need to be durable to organisational change. We have started to use 'collection' oriented URI with the existing sectored domain of environment.data.gov.uk.

The major retained use of the location.data.gov.uk is to as a vocabulary hub for vocabulary derived from INSPIRE application schema.

4.2.1 URI Patterns for INSPIRE derived Vocabulary Terms

The approach we have adopted for generating RDF/OWL vocabulary terms from INSPIRE data specifications is to follow our understanding of the spirit of ISO 19150-2. Our mapping is of the general form:

From ISO Construct	To OWL Construct	URI Pattern
Application Schema	OWL Ontology	*For vocabularies derived from INSPIRE data specifications:* ***{ontologyName} =*** **http://location.data.gov.uk/inspire/{theme}/def[/{umlPackageName}]** "inspire/{theme}" serves the role of a {/collection*} name while {umlPackageName} serves the role of {vocabulary} For vocabularies derived from local application schema **{ontologyName} =** **http://{host}[/{collection}*]/def/{umlPackageName}** Such schema arise where there is a need to specialise or extend the common INSPIRE application schema.
Feature Type	OWL Class	***{ontologyName}('#'|'/'){umlClassName}*** *As well as defining a class, a common practice to also define an open domained property for making artbitrary references to instances of that class. e.g. in the case of the Transport Network RailwayStationNode and RailwayStationArea UML feature types lead to the definition of the following classes and properties:* ***tn:RailwayStationNode a owl:Class .*** ***tn:railwayStationNode a owl:ObjectProperty ;*** ***rdfs:range tn:RailwayStationNode .*** ***tn:RailwayStationArea a owl:Class .*** ***tn:railwayStationArea a owl:ObjectProperty ;*** ***rdfs:range tn:RailwayStationArea .***
Association Association-Role	OWL ObjectProperty	***{ontologyName}('#'|'/')[{umlClassName}.]{umlAssociationRoleName}***
Attribute	OWL Object or Datatype Property	***{ontologyName}('#'|'/')[{umlClassName}.]{umlAttributeName}***

Codelist	SKOS ConceptScheme, SKOS Concept subclass, and instances	***{ontologyName}('#'|'/'){umlClassName}.scheme***
		{ontologyName}('#'|'/'){umlClassName}
		{ontologyName}('#'|'/')[{umlClassName}.[cp.]]{umlCodePointName}
		Note ISO 19150-2 is a work in progress. It does not currently dictate a particular way to distinguish scheme from class from codepoint.

Enumeration	An OWL oneOf enumerated data range.	***{ontologyName}#{umlClassName}*** ***a rdfs:Datatype*** ***owl:oneOf ("{enum1}" "{enum2}" ... "{enumN}") .*** *Where {enum1} to {enumN} correspond to the literal names of the members of the enumeration.*

Note: ISO 19150-2 has a mild bias toward the use of *'#'* as a separator between the local part of a URI and its remainder. Both '#' and '/' are generally acceptable separators for this purpose - hence *('#' | '/')* in the patterns above. Our local preference is more often to use *'/'* than *'#'.*

We follow common lexical conventions for class and property local names.

- Class local names uses camel case with an uppercase leading letter;
- Property local names use camel case with a lowercase leading letter.

- Instance local names are typically all lower case with hyphen, '-', separated words rather than camel case.

Note: for attributes and associations the approach of qualifying an attribute name or association role name with the name of the UML class for which its use is defined is there to disambiguate multiple local uses of the same attribute/association role name by different classes within the same package (i.e. ontology). Where a given property/role name is either used only once within a package or where all uses within a package are consistent in the sense of having a common range constraint (or a range constraint that can be generalised) then such qualification is unnecessary.

Note: application schema can import other application schema and/or elements of the ISO 19xxx series Harmonised Model. The package names associated with properties are the package name/application-schema name, is the one in which the property is defined. In order to share property use across the widest possible scope they should be introduced at a sufficiently high level within the combined schema's class hierarchy.

INSPIRE data-specifications use UML stereotyping on UML classes to distinguish <*featureType*> from <*datatype*>. The key difference seems to be that instances of a <*featureType*> UML Class are spatial-objects that have identity in the form of an INSPIRE spatial-object identifier. Whereas <*datatype*> although also presented as UML Classes have instances that are more like structured literals - their identity is their value - there is no means to reference them other than by value or indirectly as part of an object.

Largely we treat <*datatype*> UML classes in the same way as <*featureType*> UML classes when creating a model. However when creating instance data, we assign spatial-object identifiers (which could be as vestigial as being just a URI) for spatial-objects but for datatype instances we either use a b-node or will syntactically extend the related spatial object's identifier with the localname of the the referencing property/attribute. The following listing illustrates this pattern in the context of URI assigned of inspire spatial-object identifiers whose UML class has the stereotype <datatype>. Note this example uses the spatial-objects as nodes approach described above (see 3.2.2):

```
@prefix gcm-geo: <http://location.data.gov.uk/inspire/gcm/def/geometry/> .
@prefix foaf:  <http://xmlns.com/foaf/0.1/> .
@prefix def-sp: <http://environment.data.gov.uk/water-quality/def/sampling-point/> .
@prefix sr:    <http://data.ordnancesurvey.co.uk/ontology/spatialrelations/> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix api:   <http://purl.org/linked-data/api/vocab#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix gcm-core: <http://location.data.gov.uk/inspire/gcm/def/core/> .
@prefix version: <http://purl.org/linked-data/version#> .

<http://environment.data.gov.uk/water-quality/so/sampling-point/AN-70833004>
        gcm-core:inspireId      <http://environment.data.gov.uk/water-quality/so/sampling-point/AN-70833004/inspireId> ;
        version:currentVersion  <http://environment.data.gov.uk/water-quality/so/sampling-point/AN-70833004:1> .

<http://environment.data.gov.uk/water-quality/so/sampling-point/AN-70833004:1>
        a                            version:Version , def-sp:SamplingPoint ;
        gcm-core:inspireId           <http://environment.data.gov.uk/water-quality/so/sampling-point/AN-70833004:1/inspireId>
;
        gcm-geo:representativePoint  <http://environment.data.gov.uk/water-quality/geometry/point/epsg:27700/517600/416500> .

<http://environment.data.gov.uk/water-quality/so/sampling-point/AN-70833004/inspireId>
        a                      gcm-core:Identifier ;
        gcm-core:localId       "AN-70833004" ;
        gcm-core:namespaceId   "http://environment.data.gov.uk/water-quality/so/sampling-point/" .

<http://environment.data.gov.uk/water-quality/so/sampling-point/AN-70833004:1/inspireId>
        a                      gcm-core:Identifier ;
        gcm-core:localId       "AN-70833004" ;
        gcm-core:namespaceId   "http://environment.data.gov.uk/water-quality/so/sampling-point/" ;
        gcm-core:versionId     "1" .

<http://environment.data.gov.uk/water-quality/geometry/point/epsg:27700/517600/416500>
        gcm-geo:dimensions  "2" ;
        gcm-geo:x           "517600"^^xsd:decimal ;
        gcm-geo:y           "416500"^^xsd:decimal .
```

Note the way in which we generate a URI for the representative point, which could potentially be referenced by multiple entities. The point is effectively a structured literal. We have packed it the characteristics that define the point into its URI such that the URI itself serves as a structured literal and its structure is then elaborated in the graph. It is possible to create a service that simply serves up point data like this - indeed it is somewhat circular since all the essential information is in the request URI in the first place. However, when publishing in a triple/quad store it is also useful to have (a copy of) the materialised data in the store for the purposes of being able to make queries.

4.2.2 Transforming Data

In our we have not generally started from data that has already been prepared for publication as INSPIRE spatial-objects. Generally we are faced with tabulated data in one or more tables with more or less denormalisation (ie. entities whose properties are expanded in-line within a table that should be normalised away into a separate table and referenced from there). In particular, while the entities may be coded, they typically do not have pre-determined spatial-object identifiers. Instead, we formulate spatial-object identifiers from key identifiers within the data which we are presented. For example, the sampling point URI from the extract above:

```
http://environment.data.gov.uk/water-quality/so/sampling-point/AN-70833004
http://environment.data.gov.uk/water-quality/so/sampling-point/AN-70833004:1
```

are formulated as follows:

- The INSPIRE *namespaceId*, `http://environment.data.gov.uk/water-quality/so/sampling-point/` aligns with the revised UK approach to persistent URI, in this case within the `water-quality` collection under the `environment.data.gov.uk` domain. We have chosen to collect spatial-objects within the collection under `../so/..` to keep them separate form any locally defined supporting vocabulary (under `../def/..`) related observational/transactional datasets (under `../data/..`) and other reference items (under `../id/..` and `../doc/..`). sampling-point reflects the entity type of the spatial-object and also, maybe more significantly disambiguates the *localId* part for these objects from those of any other objects that have syntactically similar *localId* where one could not be certain of otherwise avoiding clashing URI assignments (or indeed INSPIRE object id assignments).

- The INSPIRE *localId*, in this case `AN-70833004,` is formulate from one or more fields in the source data table that can be used to produce a locally unique (within the data table) identifier - and one that is durable across repetition of the conversion process. In this particular case the sample point table is an aggregation of 7-8 regional tables each of whom have their own approach to the assignment of identifiers to sampling points. In the case of this example, 'AN' is reflective of the sampling-point having been defined by the Environment Agency's Anglian region and `70833004` is the code assigned to the sampling point by Anglian Region. Together these two fields from the source data table provide sufficient uniqueness to formulate an INSPIRE *localId*

- The INSPIRE *versionId* field, in this case `':1'` is something of a 'dummy' since we are currently working with only a single version of the spatial-object. However, we anticipate using a fields such a modified date of a table record if available or a combination of a modified date and an ordinal sequence number (within date) if more than one update is possible on a given date.

For spatial-object type (aka *featureType*) we seek to align this the derived INSPIRE spatial-object type that 'in-our-opinion' is most relevant to the entity type recorded in the table we are working with. This usually a matter of judgement. It is also usually the case that we have more information to publish than is catered for by the INSPIRE schema, which leads to specialisation (see 5.4). It is also the case that we are unable to populated all of the attributes/roles specified by a given spatial-object type.

We transform textual naming fields to *rdfs:label* and *skos:prefLabel* statements - about the spatial-object (objects as nodes).

Simple attribute values (numbers, dates, strings) are aligned (as a matter of judgement) with from the relevant INSPIRE data specification or with attributes created by a specialisation. These become simple RDF statements made using the given value (typically expressed as an XML Schema Datatyped literal or as a language tagged string).

For more complex 'structured' literals - which likely arise from UML classes that have a datatype stereotype, we typically take one of two approaches.

- We view the value as closely associated with the spatial-object and unlikely to be useful for another spatial object. In this case we extend spatial-object URI with the local name of the attribute/property and optionally discriminator, usually a small ordinal number, in the case where there may usefully be multiple values for the given attribute

```
        {spatial-object-uri}/{propertyLocalName}[/{discriminator}]
```

- We view the value as potentially sharable in the sense that other objects may have attributes that share the same value (e.g. a point position, a measurement (value and unit-of-measure)). In this case we construct a URI for the structured value from its value, for example the URI for point positions in the example above:

  ```
  http://environment.data.gov.uk/water-quality/geometry/point/epsg:27700/517600/416500
  ```

  follows the pattern:

  ```
  http://environment.data.gov.uk/water-quality/geometry/point/{srs}/{x-value}/{y-value}
  ```

  By preference we would use URI for a point position service that 'minted' URI for all possible point positions and possibly provided point translation services eg:

  ```
  http://environment.data.gov.uk/water-
  quality/geometry/point/epsg:27700/517600/416500?to-srs=epsg:6326
  ```

  Under a scheme like this it would be conceivable to create common URI for simple geometries, eg. circles or spheres, envelopes or bounding boxes.

For associations between spatial-objects, then property selection is identical to property selection for attributes however, the property values is a URI for the referenced spatial-object. It may be possible to directly construct the target URI from the data available in the source table. Alternatively there may be sufficient data available in making the reference (for example a value for a key property) that the reference can be reconciled against existing data target object.

5 Deriving Vocabulary from INSPIRE Data Specifications

The previous section has describe the general patterns we have use to create URI for RDF/OWL entities derived from artefacts in UML data specifications. Broadly we have developed vocabulary on an as-needed basis. We have not necessarily covered all the elements of a given application-schema, rather just those that we have need for a given project. As will be evident from the previous section or practices follow similar lines to those being developed for ISO 19150-2.

UML Artefact	**OWL Artefact**	**Description**
Application Schema	Ontology	
featureType UML Class	OWL Class	
dataType UML Class	OWL Class	
Attribute	OWL Datatype Property or OWL ObjectProperty	We typically use *rdfs:label*, *skos:prefLabel* and *skos:altLabel*, *dct:description* and/or *rdfs:comment* in place of properties whose general nature is to label, describe or comment. We tend to leave property domains open and restrict usage in class definitions, particularly where cardinality constraints are required. On rare occassions we may use a class specific narrowing range restriction (ie. one which reduces the number of entities available in the property range when used with that class) For structured attribute values that can be expressed as one of the build-in XSD datatypes attribute values will be transformed to *owl:DatatypeProperty*. Attributes whose values require more structure and association-roles are transformed as *owl:ObjectProperty*
Association Role	OWL Object Property	

UML Artefact	OWL Artefact	Description
Codelist	SKOS Concept Scheme SKOS Concept subClass and instances	Codelist and controlled vocabularies are transformed into SKOS concept schemes. All code points are made members of the scheme using skos:inScheme. They are also made instances of a distinguished subclass of skos:Concept which can be use to restrict the range of a property. Typically we also define an open domained property that can be used to make use of the code with an arbitrary entity.
Enumeration	An owl:oneOf enumerated data range.	We have yet to use this form in practice. It is more likely that we'd treat enumerations in a similar way to codelists.

Throughout the following sections we include the source of vocabularies mentioned as embedded objects.

5.1 Foundation Schema

So far we have only had to deal with very simple geometries, specifically points and envelopes. We have deployed some polygons as GML literals following the style adopted by the Ordnance Survey. The distinction between a feature (aka INSPIRE spatial-object) and a geometry is well worth being maintained, even for point geometries. In earlier work we have tended to apply say, *geo:lat* and *geo:long* values directly to the entities being described e.g., schools, stations, sampling points etc. However, it is much cleaner to follow the form of saying that the entity has a location or a representative position or an envelope or indeed more generally a geometry and then giving separate account of the geometry (separate in the sense that it is a distinct node, not in the sense that it can only be obtained separately).

The current state of the art in expressing feature geometries in the linked-data world is at the very least diverse and arguably confused. In large part the link-data community is waiting for a dominant common practice to arise for expressing anything more complex than a WGS84 point position using the W3C Geo vocabulary[8].

There appear to be two strands of work at OGC, one centred around GeoSPARQL [9] and the other seems to be centred around the expression of the ISO 19xxx Harmonised Model in OWL[10], work led by Simon Cox which provides a geometry vocabulary derived from ISO19107[11]. It is not clear (to thios author) what the status of this latter work is in OGC and whether or when it will migrate to an OGC hosted site.

Within the linked data community there is a community activity to specify vocabulary to support the publication of geospatial information known as NeoGeo[12].

In addition several linked data publishers use idiosyncratic means to publish, particular, geometries.

In March 2014 the W3C organised the "Linked Geospatial Data 2014" workshop, one outcome of which was a strong community desire to converge on common practice for representing geometries. It remains to be seen what joint OGC/W3C work comes to be chartered.

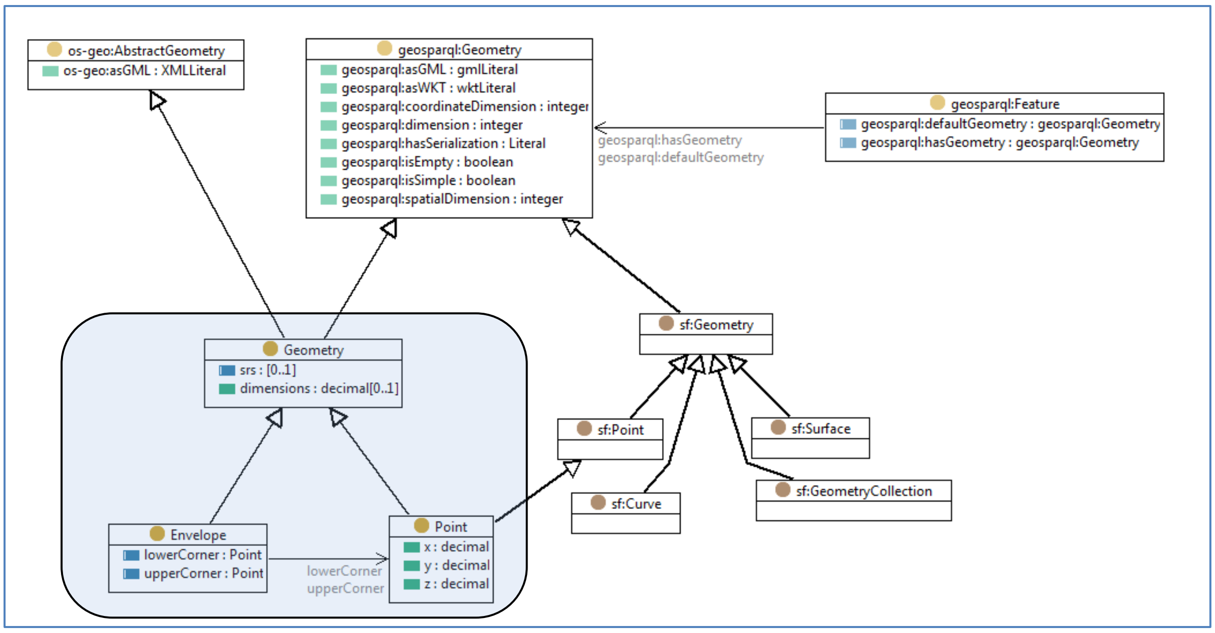
Our work initial on publishing spatial-objects as linked data pre-dated most, if not all of these initiatives.

In order to publish polygon geometries, we followed an Ordnance Survey practice of publishing GML literals (the so-call 'big-literals' approach) which is very close to the practice adopted by GeoSPARQL.

For simple bounding boxes and generalised point geometries we developed a small vocabulary, http://location.data.gov.uk/inspire/gcm/def/geometry/, illustrated below. We have latterly made our Geometry class a subclass of similarly named Classes from: the GeoSPARQL vocabulary; and the

[8] http://www.w3.org/2003/01/geo/
[9] http://www.opengis.net/ont/geosparql#
[10] http://def.seegrid.csiro.au/static/isotc211/
[11] http://def.seegrid.csiro.au/static/isotc211/iso19107/2003/geometry.ttl
[12] http://geovocab.org/

Ordnance Survey geometry ontology[13] - thus, for example they may all carry a corresponding GML literal as the value of an *os-geo:asGML* property.



os-geo:AbstractGeometry
os-geo:asGML : XMLLiteral
geosparql:Geometry
geosparql:asGML : gmlLiteral
geosparql:asWKT : wktLiteral
geosparql:coordinateDimension : integer
geosparql:dimension : integer
geosparql:hasSerialization : Literal
geosparql:isEmpty : boolean
geosparql:isSimple : boolean
geosparql:spatialDimension : integer
geosparql:hasGeometry
geosparql:defaultGeometry
geosparql:Feature
geosparql:defaultGeometry : geosparql:Geometry
geosparql:hasGeometry : geosparql:Geometry
Geometry
srs : [0..1]
dimensions : decimal[0..1]
sf:Geometry
sf:Point
sf:Surface
sf:Curve
sf:GeometryCollection
Envelope
lowerCorner : Point
upperCorner : Point
lowerCorner
upperCorner
Point
x : decimal
y : decimal
z : decimal

geometry.ttl

5.2 INSPIRE GCM

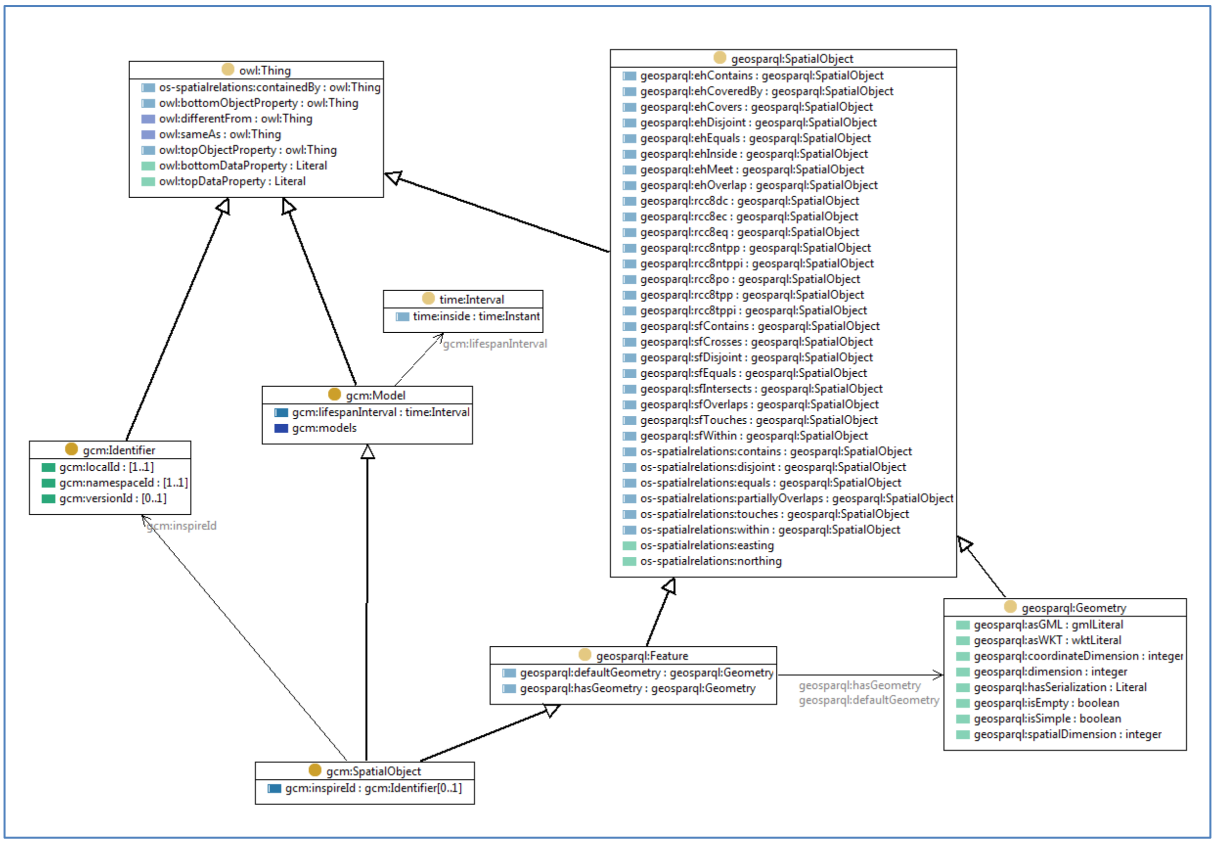
In order to represent the core concept of spatial-object and some of the based types included within the GCM we created a small core vocabulary: http://location.data.gov.uk/inspire/gcm/def/core/

core.ttl

5.2.1 gcm:Spatial Object

The following diagram is similar to the diagram introduced earlier at section 3.2.2. Here we expand a little more of classes relate to GeoSPARQL in order to emphasise that GeoSPARQL also has a notion of a Spatial Object which seems more closely aligned with the ISOTC 211 definition. Specifically that both geosparql:Feature and geosparql:Geometry are subclasses of, and hence subsumed by, *geosparql:SpatialObject*. However, *gcm:SpatialObject* is subclass of *geosparql:Feature* and specifically is not a subclass of *geosparql:Geometry*. ie. INSPIRE spatial-objects (ie. instances of *gcm:SpatialObject* or any of its subclasses) are distinct from instances of *geosparql:Geometry*.[14]

[13] http://www.ordnancesurvey.co.uk/ontology/geometry.owl
[14] Disjointness axioms are defined between geosparql:Feature and geosparql:Geometry


owl:Thing
os-spatialrelations:containedBy : owl:Thing
owl:bottomObjectProperty : owl:Thing
owl:differentFrom : owl:Thing
owl:sameAs : owl:Thing
owl:topObjectProperty : owl:Thing
owl:bottomDataProperty : Literal
owl:topDataProperty : Literal
geosparql:SpatialObject
geosparql:ehContains : geosparql:SpatialObject
geosparql:ehCoveredBy : geosparql:SpatialObject
geosparql:ehCovers : geosparql:SpatialObject
geosparql:ehDisjoint : geosparql:SpatialObject
geosparql:ehEquals : geosparql:SpatialObject
geosparql:ehInside : geosparql:SpatialObject
geosparql:ehMeet : geosparql:SpatialObject
geosparql:ehOverlap : geosparql:SpatialObject
geosparql:rcc8dc : geosparql:SpatialObject
geosparql:rcc8ec : geosparql:SpatialObject
geosparql:rcc8eq : geosparql:SpatialObject
geosparql:rcc8ntpp : geosparql:SpatialObject
geosparql:rcc8ntppi : geosparql:SpatialObject
geosparql:rcc8po : geosparql:SpatialObject
geosparql:rcc8tpp : geosparql:SpatialObject
geosparql:rcc8tppi : geosparql:SpatialObject
geosparql:sfContains : geosparql:SpatialObject
geosparql:sfCrosses : geosparql:SpatialObject
geosparql:sfDisjoint : geosparql:SpatialObject
geosparql:sfEquals : geosparql:SpatialObject
geosparql:sfIntersects : geosparql:SpatialObject
geosparql:sfOverlaps : geosparql:SpatialObject
geosparql:sfTouches : geosparql:SpatialObject
geosparql:sfWithin : geosparql:SpatialObject
os-spatialrelations:contains : geosparql:SpatialObject
os-spatialrelations:disjoint : geosparql:SpatialObject
os-spatialrelations:equals : geosparql:SpatialObject
os-spatialrelations:partiallyOverlaps : geosparql:SpatialObject
os-spatialrelations:touches : geosparql:SpatialObject
os-spatialrelations:within : geosparql:SpatialObject
os-spatialrelations:easting
os-spatialrelations:northing
time:Interval
time:inside : time:Instant
gcm:lifespanInterval
gcm:Model
gcm:lifespanInterval : time:Interval
gcm:models
gcm:Identifier
gcm:localId : [1..1]
gcm:namespaceId : [1..1]
gcm:versionId : [0..1]
gcm:inspireId
geosparql:Geometry
geosparql:asGML : gmlLiteral
geosparql:asWKT : wktLiteral
geosparql:coordinateDimension : integer
geosparql:dimension : integer
geosparql:hasSerialization : Literal
geosparql:isEmpty : boolean
geosparql:isSimple : boolean
geosparql:spatialDimension : integer
geosparql:Feature
geosparql:defaultGeometry : geosparql:Geometry
geosparql:hasGeometry : geosparql:Geometry
geosparql:hasGeometry
geosparql:defaultGeometry
gcm:SpatialObject
gcm:inspireId : gcm:Identifier[0..1]

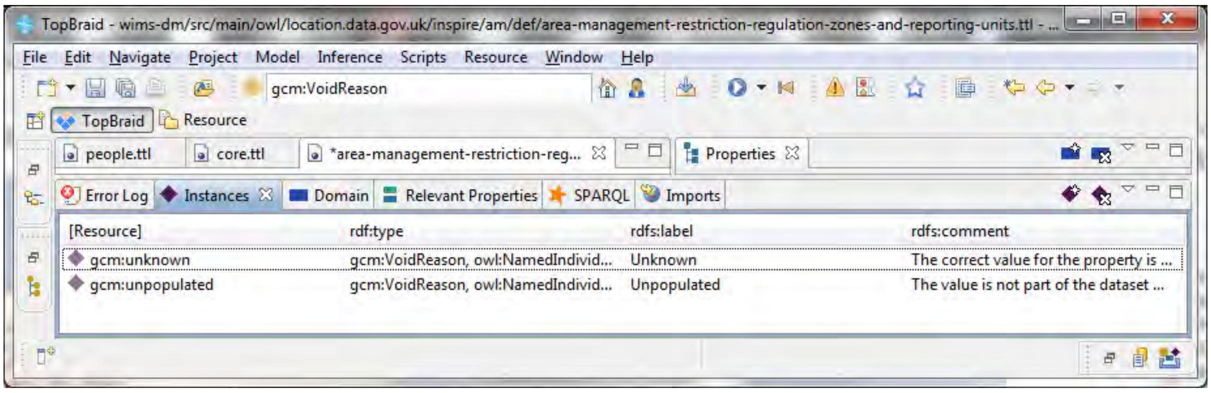
gcm:SpatialObject is motivated from two reasons. Firstly, it gives a super-class for all spatial objects and therefore provides a common home for gcm:inspireId which is defined to be reused throughout rather than redefined individually for each application-schema or spatial-object subtype. Secondly it is defined as a subclass of *gcm:Model* as described earlier, which emphasises the distinction between a model/abstraction of some-thing and the modelled-thing itself. In that respect it also serves as good carrier for lifespan-information. Some thought needs to be given to the use of lifespan information with versioned spatial objects. There are several lifespans to be covered:

- the lifespan of the real-world thing itself, which can only really be closed when it ceases to exist
- the lifespan of the (enduring) spatial-object which is potentially quite different from the lifespan of the corresponding real world thing
- the lifespan of a spatial-object version (a snapshot). However that is the role of the *version:interval* property in our versioning vocabulary.

5.2.2 Voidable Properties

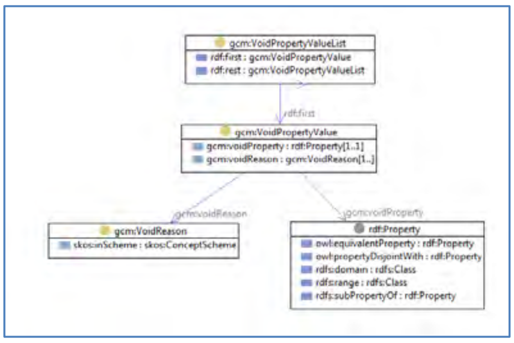
Voidable properties present a certain amount of difficulty to RDF under the open-world assumption. Just because the value of a property may not be given does not mean that there is no value for that property that could be given elsewhere. There are two possible approaches, both of which rely on the creation of a codelist for void reasons (a *skos:ConceptScheme* instance *gcm:voidReasonConceptScheme* and a *skos:Concept* subclass *gcm:VoidReason).*

1. On voidable attributes, specify an *rdfs:range* that is an *owl:unionOf* the properties 'natural' range and *gcm:VoidReason*.
2. Specify voidable properties with their 'natural' *rdfs:range* and add the facility to associated a list of void property values with instance data which state voided properties for that instance and the associated *gcm:VoidReason.*


TopBraid - wims-dm/src/main/owl/location.data.gov.uk/inspire/am/def/area-management-restriction-regulation-zones-and-reporting-units.ttl - ...
File Edit Navigate Project Model Inference Scripts Resource Window Help
gcm:VoidReason
TopBraid
Resource
people.ttl
core.ttl
*area-management-restriction-reg...
Properties
Error Log
Instances
Domain
Relevant Properties
SPARQL
Imports

[Resource]	rdf:type	rdfs:label	rdfs:comment
gcm:unknown	gcm:VoidReason, owl:NamedIndivid...	Unknown	The correct value for the property is ...
gcm:unpopulated	gcm:VoidReason, owl:NamedIndivid...	Unpopulated	The value is not part of the dataset ...

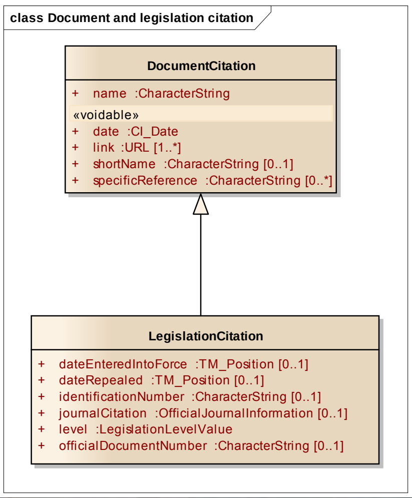
We have adopted this second approach. This includes the definition of an open-domained property for referencing *gcm:VoidPropertyValueList*, ie. *gcm:voidPropertyValueList*.

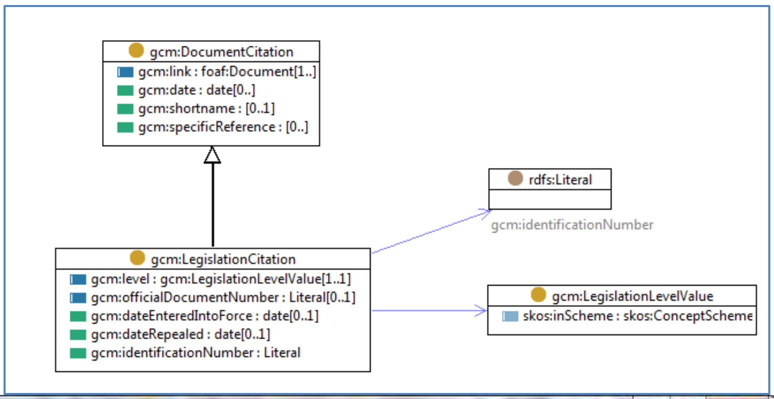

gcm:VoidPropertyValueList
rdf:first : gcm:VoidPropertyValue
rdf:rest : gcm:VoidPropertyValueList
rdf:first
gcm:VoidPropertyValue
gcm:voidProperty : rdf:Property[1..1]
gcm:voidReason : gcm:VoidReason[1..]
gcm:voidReason
gcm:voidProperty
gcm:VoidReason
skos:inScheme : skos:ConceptScheme
rdf:Property
owl:equivalentProperty : rdf:Property
owl:propertyDisjointWith : rdf:Property
rdfs:domain : rdfs:Class
rdfs:range : rdfs:Class
rdfs:subPropertyOf : rdf:Property

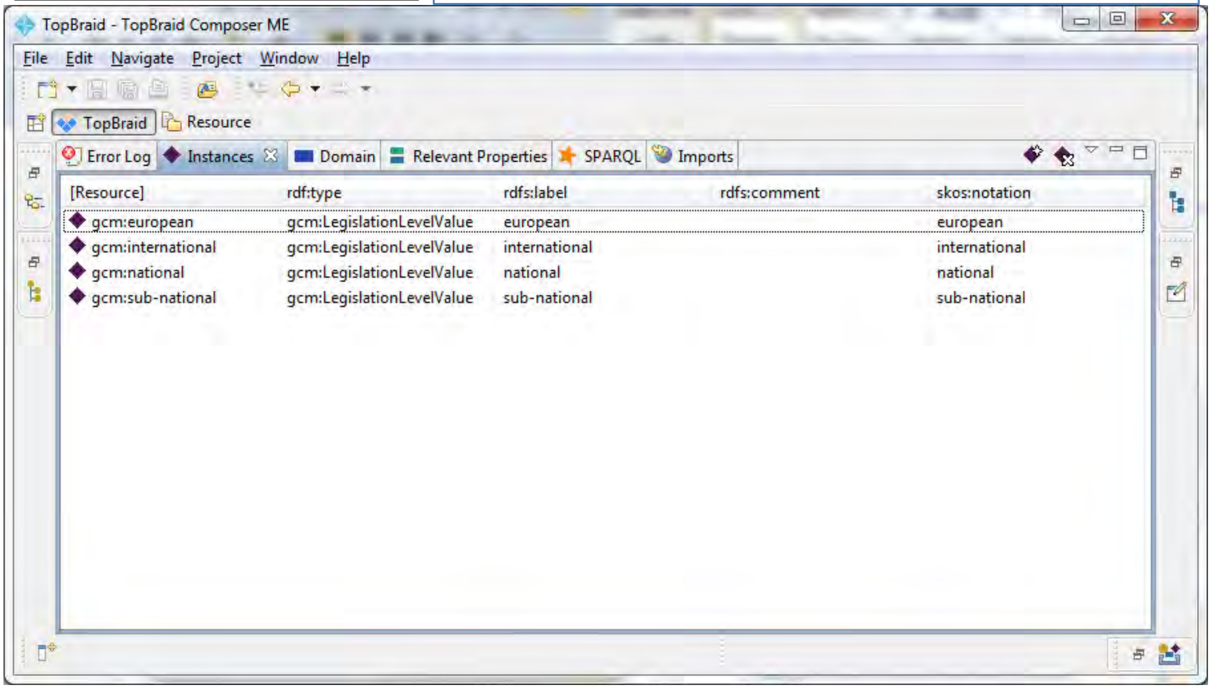
5.2.3 Legislation and Document Citations

For the Base Types, *DocumentCitation* and *LegislationCitation* we largely clone the class and field names into their OWL expression. Note that neither of these classes are stereotyped at all, specifically they are not stereotyped as either featureType or datatype. In that respect they are not made as subclasses of *gcm:SpatialObject* and are not expected to carry an *gcm:inspiredId*.

We have omitted 'journalCitation' from our transformation. 'name's assumed to map to one of the common labelling properties (*rdfs:label*, *skos:prefLabel*).


class Document and legislation citation
DocumentCitation
+ name :CharacterString
«voidable»
+ date :CI_Date
+ link :URL [1..*]
+ shortName :CharacterString [0..1]
+ specificReference :CharacterString [0..*]
LegislationCitation
+ dateEnteredIntoForce :TM_Position [0..1]
+ dateRepealed :TM_Position [0..1]
+ identificationNumber :CharacterString [0..1]
+ journalCitation :OfficialJournalInformation [0..1]
+ level :LegislationLevelValue
+ officialDocumentNumber :CharacterString [0..1]


gcm:DocumentCitation
gcm:link : foaf:Document[1..]
gcm:date : date[0..]
gcm:shortname : [0..1]
gcm:specificReference : [0..]
rdfs:Literal
gcm:identificationNumber
gcm:LegislationCitation
gcm:level : gcm:LegislationLevelValue[1..1]
gcm:officialDocumentNumber : Literal[0..1]
gcm:dateEnteredIntoForce : date[0..1]
gcm:dateRepealed : date[0..1]
gcm:identificationNumber : Literal
gcm:LegislationLevelValue
skos:inScheme : skos:ConceptScheme

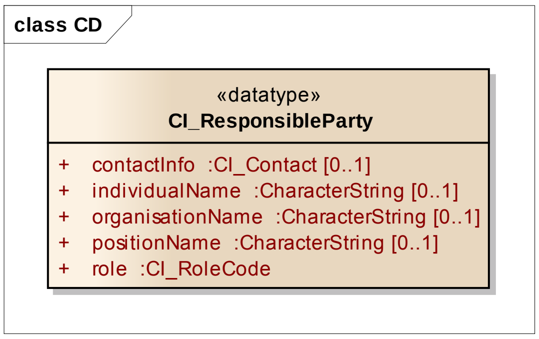


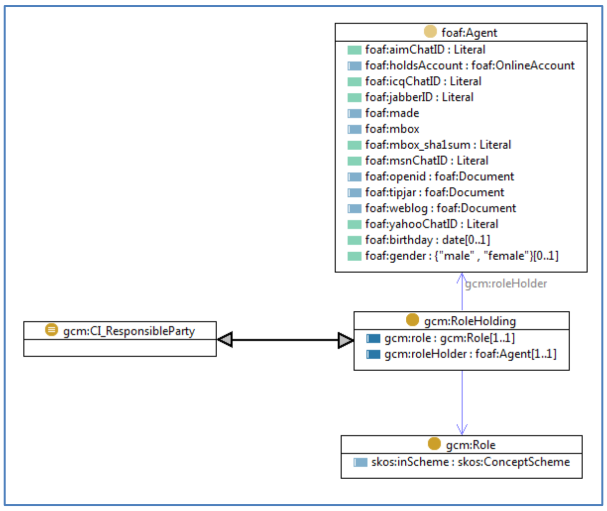
[Resource]	rdf:type	rdfs:label	rdfs:comment	skos:notation
gcm:european	gcm:LegislationLevelValue	european		european
gcm:international	gcm:LegislationLevelValue	international		international
gcm:national	gcm:LegislationLevelValue	national		national
gcm:sub-national	gcm:LegislationLevelValue	sub-national		sub-national

5.2.4 CI_ResponsibleParty

Linked-data representations of people and organisations are typically richer than the expressions that seem possible with *CI_ResponsibleParty*. The FOAF, ORG and vCard are vocabularies that are commonly used in the linked-data community for representing people, roles and organisational structure. The ISA core vocabularies may also provides some coverage of this space too.

For CI_ResponsibleParty we have tried to make more use of FOAF as a way of representing information about a person or organization. We have defined OWL classes for *gcm:CI_ResponsibleParty* and *gcm:RoleHolding*. *gcm:RoleHolding* essentially reifies the relation between a *foaf:Agent* and a coded role. The same *foaf:Agent* may have many roles in different contexts and having reified the holding of the role, information about the interval over which the role holder was active in the role could be added.

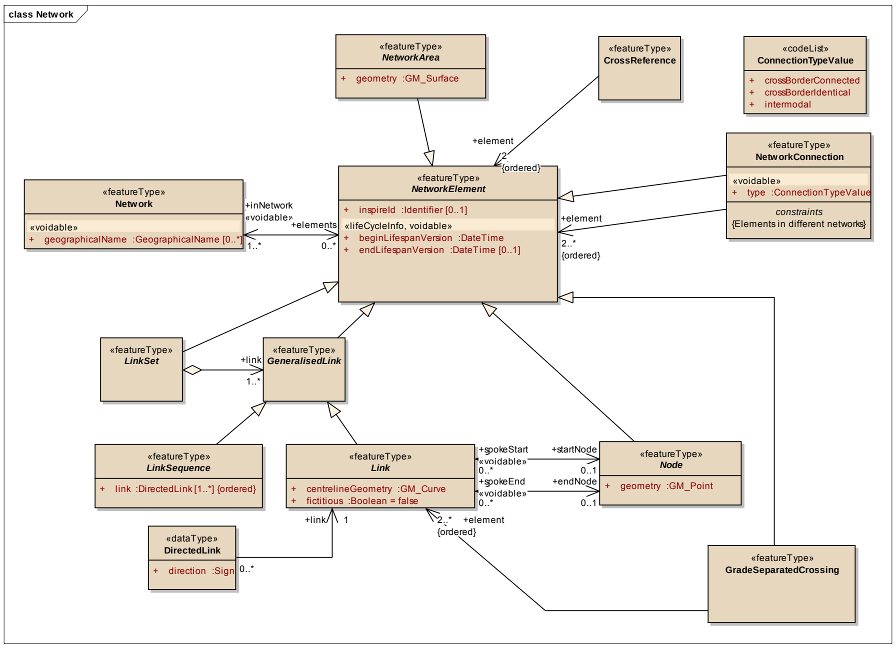

class CD
«datatype»
CI_ResponsibleParty
+ contactInfo :CI_Contact [0..1]
+ individualName :CharacterString [0..1]
+ organisationName :CharacterString [0..1]
+ positionName :CharacterString [0..1]
+ role :CI_RoleCode

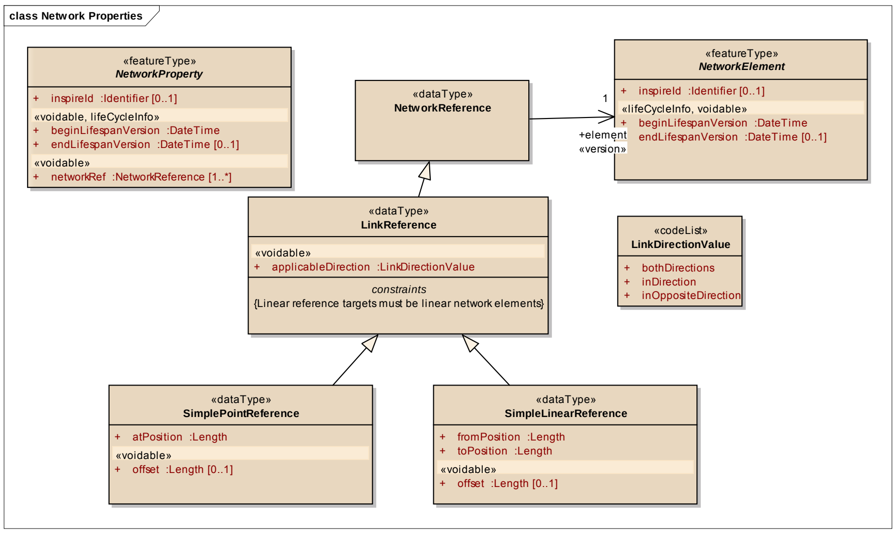

foaf:Agent
foaf:aimChatID : Literal
foaf:holdsAccount : foaf:OnlineAccount
foaf:icqChatID : Literal
foaf:jabberID : Literal
foaf:made
foaf:mbox
foaf:mbox_sha1sum : Literal
foaf:msnChatID : Literal
foaf:openid : foaf:Document
foaf:tipjar : foaf:Document
foaf:weblog : foaf:Document
foaf:yahooChatID : Literal
foaf:birthday : date[0..1]
foaf:gender : {"male" , "female"}[0..1]
gcm:roleHolder
gcm:CI_ResponsibleParty
gcm:RoleHolding
gcm:role : gcm:Role[1..1]
gcm:roleHolder : foaf:Agent[1..1]
gcm:Role
skos:inScheme : skos:ConceptScheme

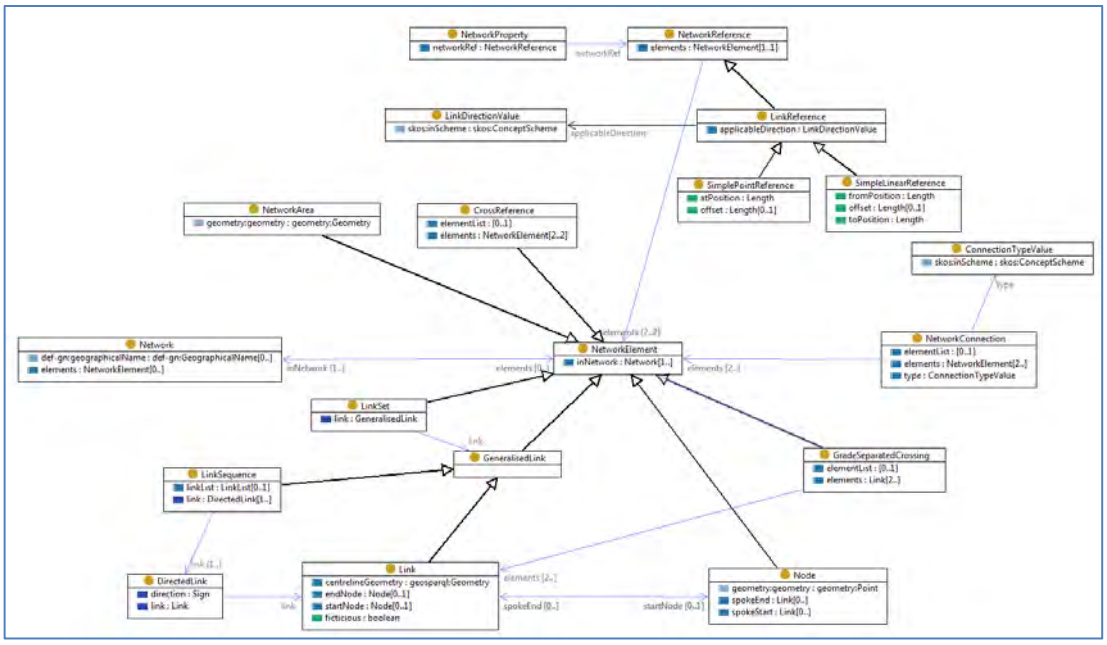
5.2.5 GCM Network Base Model

The following diagrams illustrate the derivation of the ontology http://location.data.gov.uk/inspire/gcm/def/network/ from GCM Network Base Model application schema.

network.ttl


class Network
«featureType» NetworkArea
+ geometry :GM_Surface
«featureType» CrossReference
«codeList» ConnectionTypeValue
+ crossBorderConnected
+ crossBorderIdentical
+ intermodal
«featureType» NetworkConnection
«voidable»
+ type :ConnectionTypeValue
constraints
{Elements in different networks}
«featureType» Network
«voidable»
+ geographicalName :GeographicalName [0..*]
«featureType» NetworkElement
+ inspireId :Identifier [0..1]
«lifeCycleInfo, voidable»
+ beginLifespanVersion :DateTime
+ endLifespanVersion :DateTime [0..1]
+inNetwork
«voidable»
+elements
1..*
0..*
+element
2
{ordered}
+element
2..*
{ordered}
«featureType» LinkSet
«featureType» GeneralisedLink
+link
1..*
«featureType» LinkSequence
+ link :DirectedLink [1..*] {ordered}
«featureType» Link
+ centrelineGeometry :GM_Curve
+ fictitious :Boolean = false
+spokeStart
«voidable»
0..*
+spokeEnd
«voidable»
0..*
+startNode
0..1
+endNode
0..1
«featureType» Node
+ geometry :GM_Point
«dataType» DirectedLink
+ direction :Sign
0..*
+link
1
2..*
+element
{ordered}
«featureType» GradeSeparatedCrossing


class Network Properties
«featureType» NetworkProperty
+ inspireId :Identifier [0..1]
«voidable, lifeCycleInfo»
+ beginLifespanVersion :DateTime
+ endLifespanVersion :DateTime [0..1]
«voidable»
+ networkRef :NetworkReference [1..*]
«dataType» NetworkReference
«featureType» NetworkElement
+ inspireId :Identifier [0..1]
«lifeCycleInfo, voidable»
+ beginLifespanVersion :DateTime
+ endLifespanVersion :DateTime [0..1]
1
+element
«version»
«dataType» LinkReference
«voidable»
+ applicableDirection :LinkDirectionValue
constraints
{Linear reference targets must be linear network elements}
«codeList» LinkDirectionValue
+ bothDirections
+ inDirection
+ inOppositeDirection
«dataType» SimplePointReference
+ atPosition :Length
«voidable»
+ offset :Length [0..1]
«dataType» SimpleLinearReference
+ fromPosition :Length
+ toPosition :Length
«voidable»
+ offset :Length [0..1]


NetworkProperty
networkRef : NetworkReference
networkRef
NetworkReference
elements : NetworkElement[1..1]
LinkDirectionValue
skos:inScheme : skos:ConceptScheme
applicableDirection
LinkReference
applicableDirection : LinkDirectionValue
SimplePointReference
atPosition : Length
offset : Length[0..1]
SimpleLinearReference
fromPosition : Length
offset : Length[0..1]
toPosition : Length
NetworkArea
geometry:geometry : geometry:Geometry
CrossReference
elementList : [0..1]
elements : NetworkElement[2..2]
ConnectionTypeValue
skos:inScheme : skos:ConceptScheme
type
Network
def-gn:geographicalName : def-gn:GeographicalName[0..]
elements : NetworkElement[0..]
inNetwork [1..]
elements [0..]
elements [2..2]
NetworkElement
inNetwork : Network[1..]
elements [2..]
NetworkConnection
elementList : [0..1]
elements : NetworkElement[2..]
type : ConnectionTypeValue
LinkSet
link : GeneralisedLink
link
GeneralisedLink
GradeSeparatedCrossing
elementList : [0..1]
elements : Link[2..]
LinkSequence
linkList : LinkList[0..1]
link : DirectedLink[1..]
link [1..]
DirectedLink
direction : Sign
link : Link
link
Link
centrelineGeometry : geosparql:Geometry
endNode : Node[0..1]
startNode : Node[0..1]
fictitious : boolean
elements [2..]
spokeEnd [0..]
startNode [0..1]
Node
geometry:geometry : geometry:Point
spokeEnd : Link[0..]
spokeStart : Link[0..]

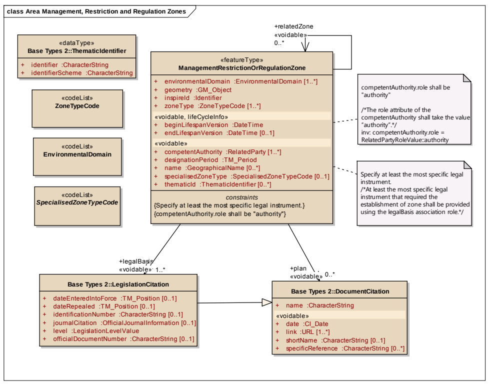
5.3 Thematic Application Schema

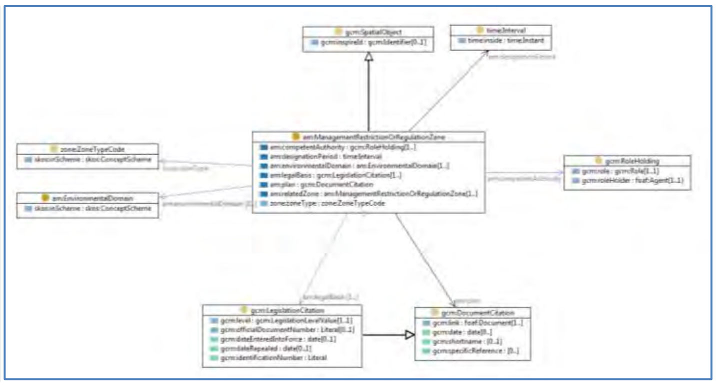
5.3.1 Area Management, Regulation and Restriction Zones

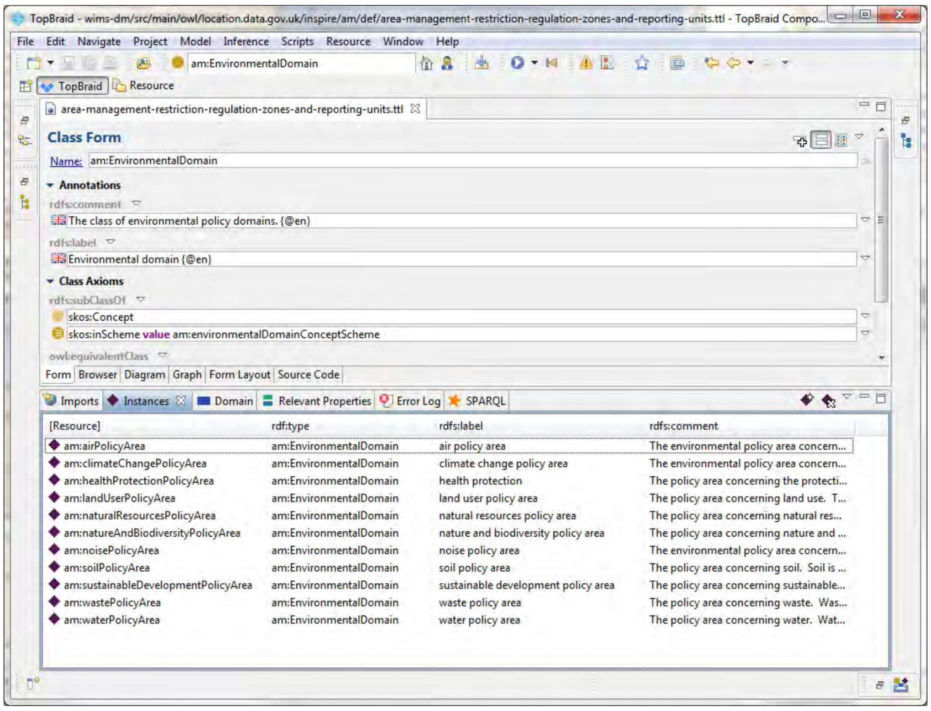
Following our URI pattern guidance we create an ontology derived from the the top-level area-management application schema at:
http://location.data.gov.uk/inspire/am/def/area-management-restriction-regulation-zones-and-reporting-units/

area-management-restriction-regulation-zones-and-reporting-units.ttl

The diagrams below illustrate the fragment of this application schema that we have implemented.


class Area Management, Restriction and Regulation Zones
«dataType»
Base Types 2::ThematicIdentifier
+ identifier :CharacterString
+ identifierScheme :CharacterString
«codeList»
ZoneTypeCode
«codeList»
EnvironmentalDomain
«codeList»
SpecialisedZoneTypeCode
+relatedZone
«voidable»
0..*
«featureType»
ManagementRestrictionOrRegulationZone
+ environmentalDomain :EnvironmentalDomain [1..*]
+ geometry :GM_Object
+ inspireId :Identifier
+ zoneType :ZoneTypeCode [1..*]
«voidable, lifeCycleInfo»
+ beginLifespanVersion :DateTime
+ endLifespanVersion :DateTime [0..1]
«voidable»
+ competentAuthority :RelatedParty [1..*]
+ designationPeriod :TM_Period
+ name :GeographicalName [0..*]
+ specialisedZoneType :SpecialisedZoneTypeCode [0..1]
+ thematicId :ThematicIdentifier [0..*]
constraints
{Specify at least the most specific legal instrument.}
{competentAuthority.role shall be "authority"}
competentAuthority.role shall be "authority"
/*The role attribute of the competentAuthority shall take the value "authority".*/
inv: competentAuthority.role = RelatedPartyRoleValue::authority
Specify at least the most specific legal instrument.
/*At least the most specific legal instrument that required the establishment of zone shall be provided using the legalBasis association role.*/
+legalBasis
«voidable» 1..*
+plan
«voidable» 0..*
Base Types 2::LegislationCitation
+ dateEnteredIntoForce :TM_Position [0..1]
+ dateRepealed :TM_Position [0..1]
+ identificationNumber :CharacterString [0..1]
+ journalCitation :OfficialJournalInformation [0..1]
+ level :LegislationLevelValue
+ officialDocumentNumber :CharacterString [0..1]
Base Types 2::DocumentCitation
+ name :CharacterString
«voidable»
+ date :CI_Date
+ link :URL [1..*]
+ shortName :CharacterString [0..1]
+ specificReference :CharacterString [0..*]


gcm:SpatialObject
gcm:inspireId : gcm:Identifier[0..1]
time:Interval
time:inside : time:Instant
am:ManagementRestrictionOrRegulationZone
am:competentAuthority : gcm:RoleHolding[1..]
am:designationPeriod : time:Interval
am:environmentalDomain : am:EnvironmentalDomain[1..]
am:legalBasis : gcm:LegislationCitation[1..]
am:plan : gcm:DocumentCitation
am:relatedZone : am:ManagementRestrictionOrRegulationZone[1..]
zone:zoneType : zone:ZoneTypeCode
zone:ZoneTypeCode
skos:inScheme : skos:ConceptScheme
am:EnvironmentalDomain
skos:inScheme : skos:ConceptScheme
gcm:RoleHolding
gcm:role : gcm:Role[1..1]
gcm:roleHolder : foaf:Agent[1..1]
gcm:LegislationCitation
gcm:level : gcm:LegislationLevelValue[1..1]
gcm:officialDocumentNumber : Literal[0..1]
gcm:dateEnteredIntoForce : date[0..1]
gcm:dateRepealed : date[0..1]
gcm:identificationNumber : Literal
gcm:DocumentCitation
gcm:link : foaf:Document[1..]
gcm:date : date[0..]
gcm:shortname : [0..1]
gcm:specificReference : [0..]



am:EnvironmentalDomain

area-management-restriction-regulation-zones-and-reporting-units.ttl

Class Form

Name: am:EnvironmentalDomain

Annotations

rdfs:comment
The class of environmental policy domains. (@en)

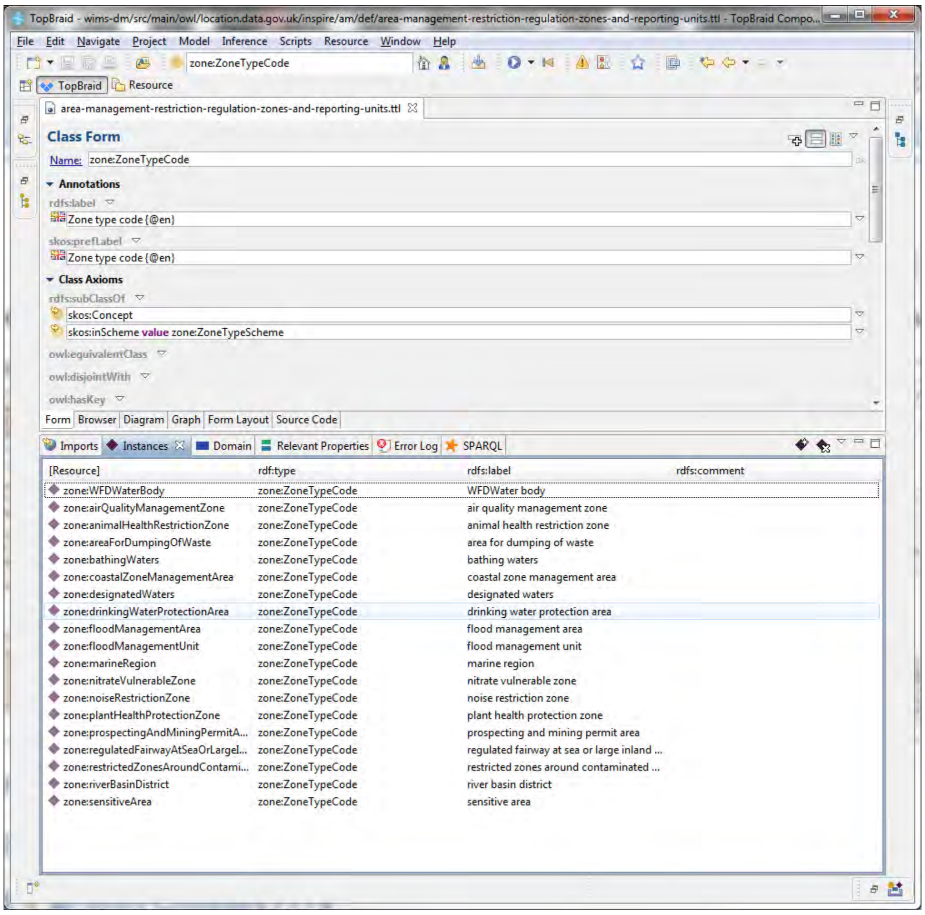
rdfs:label
Environmental domain (@en)

Class Axioms

rdfs:subClassOf
skos:Concept
skos:inScheme value am:environmentalDomainConceptScheme

owl:equivalentClass

[Resource]	rdf:type	rdfs:label	rdfs:comment
am:airPolicyArea	am:EnvironmentalDomain	air policy area	The environmental policy area concern...
am:climateChangePolicyArea	am:EnvironmentalDomain	climate change policy area	The environmental policy area concern...
am:healthProtectionPolicyArea	am:EnvironmentalDomain	health protection	The policy area concerning the protecti...
am:landUserPolicyArea	am:EnvironmentalDomain	land user policy area	The policy area concerning land use. T...
am:naturalResourcesPolicyArea	am:EnvironmentalDomain	natural resources policy area	The policy area concerning natural res...
am:natureAndBiodiversityPolicyArea	am:EnvironmentalDomain	nature and biodiversity policy area	The policy area concerning nature and ...
am:noisePolicyArea	am:EnvironmentalDomain	noise policy area	The environmental policy area concern...
am:soilPolicyArea	am:EnvironmentalDomain	soil policy area	The policy area concerning soil. Soil is ...
am:sustainableDevelopmentPolicyArea	am:EnvironmentalDomain	sustainable development policy area	The policy area concerning sustainable...
am:wastePolicyArea	am:EnvironmentalDomain	waste policy area	The policy area concerning waste. Was...
am:waterPolicyArea	am:EnvironmentalDomain	water policy area	The policy area concerning water. Wat...


TopBraid - wims-dm/src/main/owl/location.data.gov.uk/inspire/am/def/area-management-restriction-regulation-zones-and-reporting-units.ttl - TopBraid Compo...
File Edit Navigate Project Model Inference Scripts Resource Window Help
TopBraid
Resource
area-management-restriction-regulation-zones-and-reporting-units.ttl
Form Browser Diagram Graph Form Layout Source Code
Imports Instances Domain Relevant Properties Error Log SPARQL

Class Form

Name: zone:ZoneTypeCode

Annotations

rdfs:label

Zone type code {@en}

skos:prefLabel

Zone type code {@en}

Class Axioms

rdfs:subClassOf

skos:Concept

skos:inScheme **value** zone:ZoneTypeScheme

owl:equivalentClass

owl:disjointWith

owl:hasKey

[Resource]	rdf:type	rdfs:label	rdfs:comment
zone:WFDWaterBody	zone:ZoneTypeCode	WFDWater body	
zone:airQualityManagementZone	zone:ZoneTypeCode	air quality management zone	
zone:animalHealthRestrictionZone	zone:ZoneTypeCode	animal health restriction zone	
zone:areaForDumpingOfWaste	zone:ZoneTypeCode	area for dumping of waste	
zone:bathingWaters	zone:ZoneTypeCode	bathing waters	
zone:coastalZoneManagementArea	zone:ZoneTypeCode	coastal zone management area	
zone:designatedWaters	zone:ZoneTypeCode	designated waters	
zone:drinkingWaterProtectionArea	zone:ZoneTypeCode	drinking water protection area	
zone:floodManagementArea	zone:ZoneTypeCode	flood management area	
zone:floodManagementUnit	zone:ZoneTypeCode	flood management unit	
zone:marineRegion	zone:ZoneTypeCode	marine region	
zone:nitrateVulnerableZone	zone:ZoneTypeCode	nitrate vulnerable zone	
zone:noiseRestrictionZone	zone:ZoneTypeCode	noise restriction zone	
zone:plantHealthProtectionZone	zone:ZoneTypeCode	plant health protection zone	
zone:prospectingAndMiningPermitA...	zone:ZoneTypeCode	prospecting and mining permit area	
zone:regulatedFairwayAtSeaOrLargeI...	zone:ZoneTypeCode	regulated fairway at sea or large inland ...	
zone:restrictedZonesAroundContami...	zone:ZoneTypeCode	restricted zones around contaminated ...	
zone:riverBasinDistrict	zone:ZoneTypeCode	river basin district	
zone:sensitiveArea	zone:ZoneTypeCode	sensitive area	

Note that we have placed the *ZoneTypeCode* codelist in a separate 'file' in a separate namespace at: http://location.data.gov.uk/inspire/am/def/zone-type/

zone-type.ttl

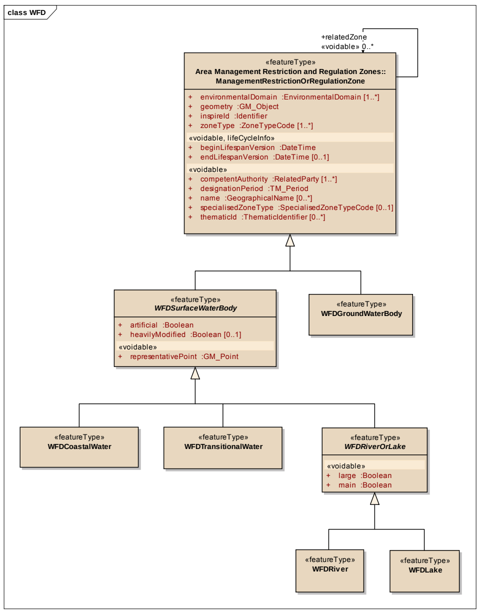
There is an interesting question here about the management of codelists and whether that should be separated from the 'schema' in which they are defined or referenced. In contrast the *EnvironmentalDomain* code list is embedded within the ontology derived from the application schema.

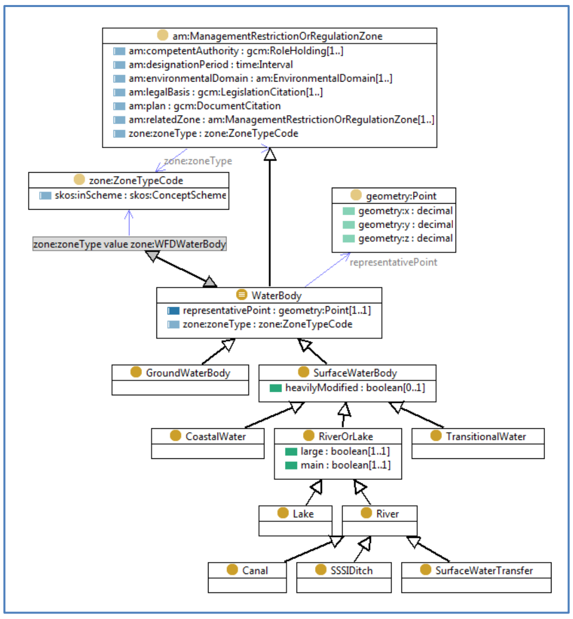
Separation makes for easier management because the codelists can be handled as a unit and changes (addition and deprecation of code points) can be performed with altering the expression of the derived ontology. However, it also tends to lead to a proliferation of namespaces which can be very tedious for codelists that either have very limited utility (ie. in practice only useful with instances of a given class) or that have very few code points.

5.3.1.1 WFD Waterbodies

Within the Administrative Areas theme there is also an application schema for spatial-objects arising under the EU Water Framework Directive (WFD). Following our pattern we have created a derived ontology with a namespace of: http://location.data.gov.uk/inspire/am/def/water-framwork-directive/

water-framework-directive.ttl

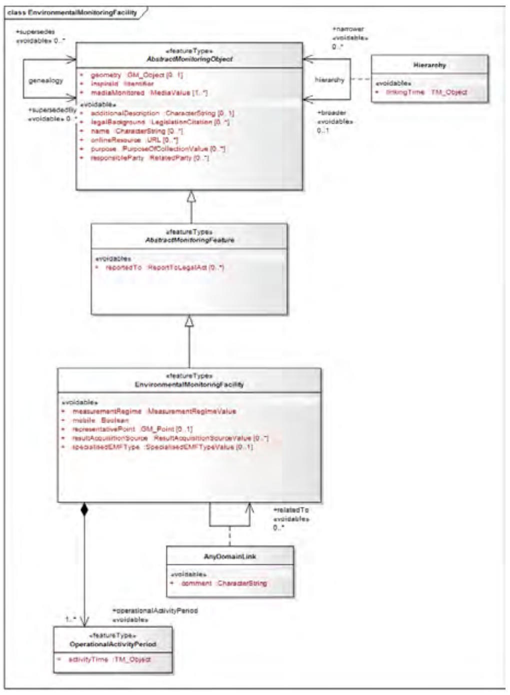

class WFD
+relatedZone
«voidable» 0..*
«featureType»
Area Management Restriction and Regulation Zones::
ManagementRestrictionOrRegulationZone
+ environmentalDomain :EnvironmentalDomain [1..*]
+ geometry :GM_Object
+ inspireId :Identifier
+ zoneType :ZoneTypeCode [1..*]
«voidable, lifeCycleInfo»
+ beginLifespanVersion :DateTime
+ endLifespanVersion :DateTime [0..1]
«voidable»
+ competentAuthority :RelatedParty [1..*]
+ designationPeriod :TM_Period
+ name :GeographicalName [0..*]
+ specialisedZoneType :SpecialisedZoneTypeCode [0..1]
+ thematicId :ThematicIdentifier [0..*]
«featureType»
WFDSurfaceWaterBody
+ artificial :Boolean
+ heavilyModified :Boolean [0..1]
«voidable»
+ representativePoint :GM_Point
«featureType»
WFDGroundWaterBody
«featureType»
WFDCoastalWater
«featureType»
WFDTransitionalWater
«featureType»
WFDRiverOrLake
«voidable»
+ large :Boolean
+ main :Boolean
«featureType»
WFDRiver
«featureType»
WFDLake

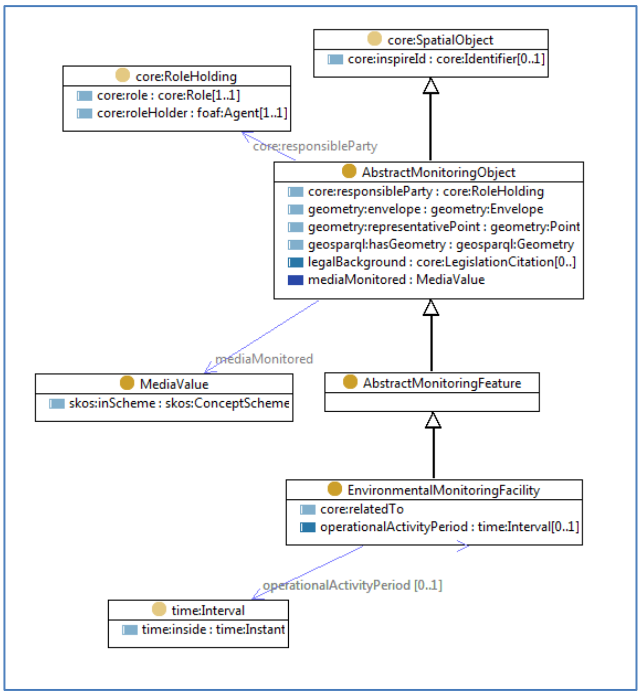

am:ManagementRestrictionOrRegulationZone
am:competentAuthority : gcm:RoleHolding[1..]
am:designationPeriod : time:Interval
am:environmentalDomain : am:EnvironmentalDomain[1..]
am:legalBasis : gcm:LegislationCitation[1..]
am:plan : gcm:DocumentCitation
am:relatedZone : am:ManagementRestrictionOrRegulationZone[1..]
zone:zoneType : zone:ZoneTypeCode
zone:zoneType
zone:ZoneTypeCode
skos:inScheme : skos:ConceptScheme
zone:zoneType value zone:WFDWaterBody
geometry:Point
geometry:x : decimal
geometry:y : decimal
geometry:z : decimal
representativePoint
WaterBody
representativePoint : geometry:Point[1..1]
zone:zoneType : zone:ZoneTypeCode
GroundWaterBody
SurfaceWaterBody
heavilyModified : boolean[0..1]
CoastalWater
RiverOrLake
large : boolean[1..1]
main : boolean[1..1]
TransitionalWater
Lake
River
Canal
SSSIDitch
SurfaceWaterTransfer

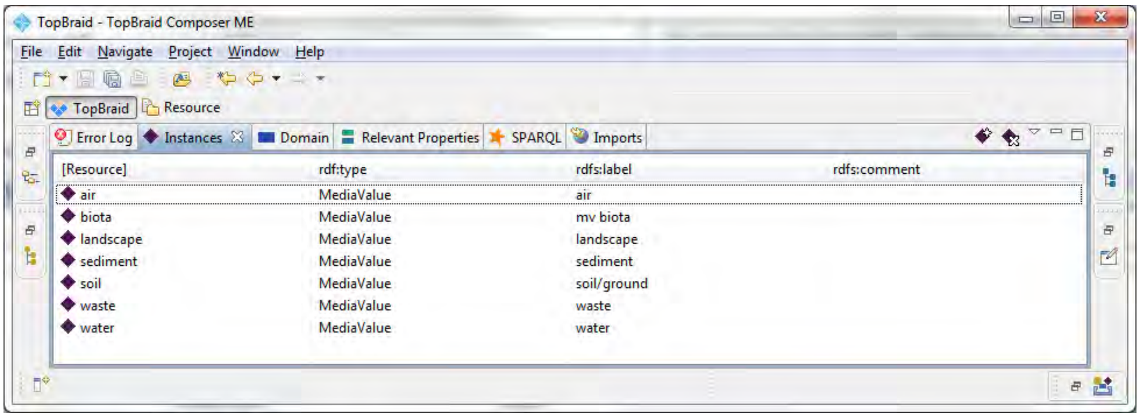
5.3.2 Environmental Monitoring Facilities

Under the Environmental Monitoring Facilities theme we have derived an ontology, illustrated below from a fragment of the environmental monitoring facilities application-schema with a namespace of: http://location.data.gov.uk/inspire/ef/def/environmental-monitoring-facilities/

environmental-monitoring-facilities.ttl


class EnvironmentalMonitoringFacility
+supersedes
«voidable» 0..*
genealogy
+supersededBy
«voidable» 0..*
«featureType»
AbstractMonitoringObject
+ geometry :GM_Object [0..1]
+ inspireId :Identifier
+ mediaMonitored :MediaValue [1..*]
«voidable»
+ additionalDescription :CharacterString [0..1]
+ legalBackground :LegislationCitation [0..*]
+ name :CharacterString [0..*]
+ onlineResource :URL [0..*]
+ purpose :PurposeOfCollectionValue [0..*]
+ responsibleParty :RelatedParty [0..*]
+narrower
«voidable»
0..*
hierarchy
+broader
«voidable»
0..1
Hierarchy
«voidable»
+ linkingTime :TM_Object
«featureType»
AbstractMonitoringFeature
«voidable»
+ reportedTo :ReportToLegalAct [0..*]
«featureType»
EnvironmentalMonitoringFacility
«voidable»
+ measurementRegime :MeasurementRegimeValue
+ mobile :Boolean
+ representativePoint :GM_Point [0..1]
+ resultAcquisitionSource :ResultAcquisitionSourceValue [0..*]
+ specialisedEMFType :SpecialisedEMFTypeValue [0..1]
+relatedTo
«voidable»
0..*
AnyDomainLink
«voidable»
+ comment :CharacterString
1..*
+operationalActivityPeriod
«voidable»
«featureType»
OperationalActivityPeriod
+ activityTime :TM_Object


core:SpatialObject
core:inspireId : core:Identifier[0..1]
core:RoleHolding
core:role : core:Role[1..1]
core:roleHolder : foaf:Agent[1..1]
core:responsibleParty
AbstractMonitoringObject
core:responsibleParty : core:RoleHolding
geometry:envelope : geometry:Envelope
geometry:representativePoint : geometry:Point
geosparql:hasGeometry : geosparql:Geometry
legalBackground : core:LegislationCitation[0..]
mediaMonitored : MediaValue
mediaMonitored
MediaValue
skos:inScheme : skos:ConceptScheme
AbstractMonitoringFeature
EnvironmentalMonitoringFacility
core:relatedTo
operationalActivityPeriod : time:Interval[0..1]
operationalActivityPeriod [0..1]
time:Interval
time:inside : time:Instant



[Resource]	rdf:type	rdfs:label	rdfs:comment
air	MediaValue	air	
biota	MediaValue	mv biota	
landscape	MediaValue	landscape	
sediment	MediaValue	sediment	
soil	MediaValue	soil/ground	
waste	MediaValue	waste	
water	MediaValue	water	

5.4 Specialisation of INSPIRE Application Schema in RDF

As part of their response to the EU Water Framework Directive, the UK Environment Agency are creating a catchment planning system. They intend to openly publish much of their catchment-planning data as linked data. Their intention at least is to publish in a form that at least follows the spirit of the INSPIRE data specification. However, in order to build the data driven applications that they want to on top of the linked-data, they need to publish much more of the data that they hold than is required by the INSPIRE data specifications. This has resulted in a need to (locally) extend the INSPIRE application schema to cover the data that EA want to publish.

As a link-data project, we have created ontologies that extend the derived ontologies discussed in the previous sections. However, we have not created UML based application schema from which to work forward through a derivation process to create derived ontologies. This raises an interesting question about 'locally' defined application schema and whether there is a need for the transformations to be bi-directional, or whether there is an obligation to always work from a UML based application schema.

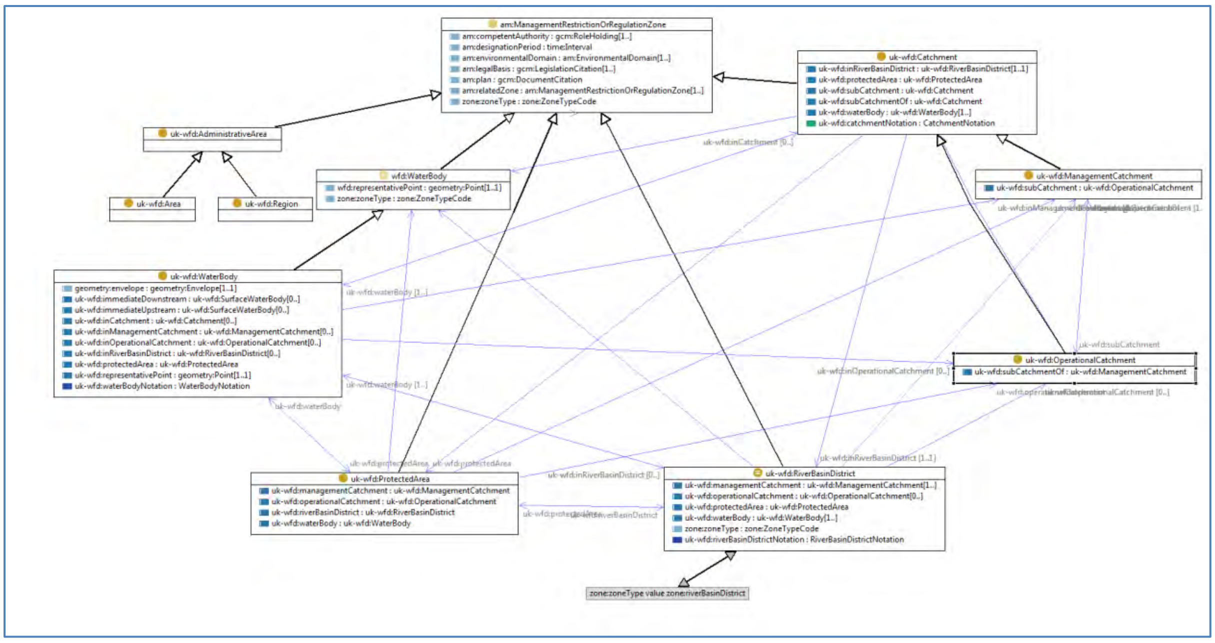
The local ontology that extends the derived INSPIRE water framework directive

http://environment.data.gov.uk/catchment-planning/def/water-framework-directive/

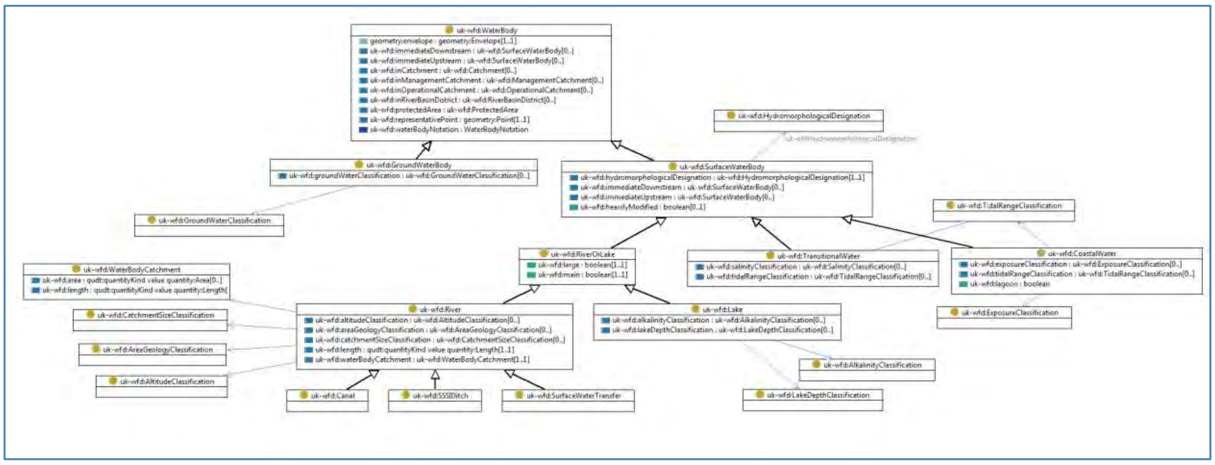
water-framework-directive.ttl

This places the extensions within the 'catchment-planning' collection of 'environment.data.gov.uk'.

The UK Environment Agency organises and administers their coverage as Regions and Area and they divide River Basin Districts into a two level structure of Catchments, Management Catchments and Operational catchments. WFD Waterbodies are associated with all of these entities, and potential with Protected Areas.


am:ManagementRestrictionOrRegulationZone
am:competentAuthority : gcm:RoleHolding[1..]
am:designationPeriod : time:Interval
am:environmentalDomain : am:EnvironmentalDomain[1..]
am:legalBasis : gcm:LegislationCitation[1..]
am:plan : gcm:DocumentCitation
am:relatedZone : am:ManagementRestrictionOrRegulationZone[1..]
zone:zoneType : zone:ZoneTypeCode
uk-wfd:Catchment
uk-wfd:inRiverBasinDistrict : uk-wfd:RiverBasinDistrict[1..1]
uk-wfd:protectedArea : uk-wfd:ProtectedArea
uk-wfd:subCatchment : uk-wfd:Catchment
uk-wfd:subCatchmentOf : uk-wfd:Catchment
uk-wfd:waterBody : uk-wfd:WaterBody[1..]
uk-wfd:catchmentNotation : CatchmentNotation
uk-wfd:AdministrativeArea
uk-wfd:Area
uk-wfd:Region
wfd:WaterBody
wfd:representativePoint : geometry:Point[1..1]
zone:zoneType : zone:ZoneTypeCode
uk-wfd:inCatchment [0..]
uk-wfd:ManagementCatchment
uk-wfd:subCatchment : uk-wfd:OperationalCatchment
uk-wfd:WaterBody
geometry:envelope : geometry:Envelope[1..1]
uk-wfd:immediateDownstream : uk-wfd:SurfaceWaterBody[0..]
uk-wfd:immediateUpstream : uk-wfd:SurfaceWaterBody[0..]
uk-wfd:inCatchment : uk-wfd:Catchment[0..]
uk-wfd:inManagementCatchment : uk-wfd:ManagementCatchment[0..]
uk-wfd:inOperationalCatchment : uk-wfd:OperationalCatchment[0..]
uk-wfd:inRiverBasinDistrict : uk-wfd:RiverBasinDistrict[0..]
uk-wfd:protectedArea : uk-wfd:ProtectedArea
uk-wfd:representativePoint : geometry:Point[1..1]
uk-wfd:waterBodyNotation : WaterBodyNotation
uk-wfd:waterBody [1..]
uk-wfd:subCatchment
uk-wfd:OperationalCatchment
uk-wfd:subCatchmentOf : uk-wfd:ManagementCatchment
uk-wfd:inOperationalCatchment [0..]
uk-wfd:waterBody
uk-wfd:protectedArea
uk-wfd:inRiverBasinDistrict [1..1]
uk-wfd:ProtectedArea
uk-wfd:managementCatchment : uk-wfd:ManagementCatchment
uk-wfd:operationalCatchment : uk-wfd:OperationalCatchment
uk-wfd:riverBasinDistrict : uk-wfd:RiverBasinDistrict
uk-wfd:waterBody : uk-wfd:WaterBody
uk-wfd:inRiverBasinDistrict [0..]
uk-wfd:RiverBasinDistrict
uk-wfd:managementCatchment : uk-wfd:ManagementCatchment[1..]
uk-wfd:operationalCatchment : uk-wfd:OperationalCatchment[0..]
uk-wfd:protectedArea : uk-wfd:ProtectedArea
uk-wfd:waterBody : uk-wfd:WaterBody[1..]
zone:zoneType : zone:ZoneTypeCode
uk-wfd:riverBasinDistrictNotation : RiverBasinDistrictNotation
zone:zoneType value zone:riverBasinDistrict

In addition UK water bodies are subject to a number of classifications depending on the type of water body. Each of these classifications is maintained as a coded list as illustrated in the diagram below.

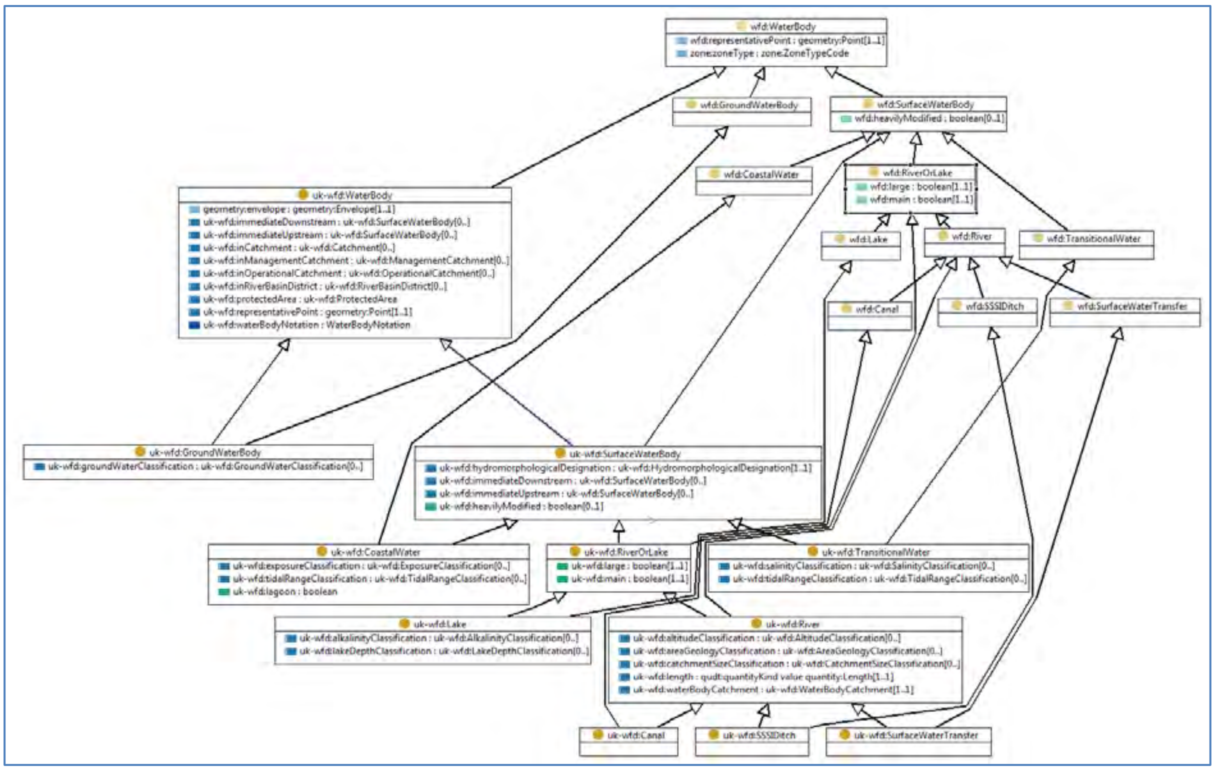

uk-wfd:WaterBody
geometry:envelope : geometry:Envelope[1..1]
uk-wfd:immediateDownstream : uk-wfd:SurfaceWaterBody[0..]
uk-wfd:immediateUpstream : uk-wfd:SurfaceWaterBody[0..]
uk-wfd:inCatchment : uk-wfd:Catchment[0..]
uk-wfd:inManagementCatchment : uk-wfd:ManagementCatchment[0..]
uk-wfd:inOperationalCatchment : uk-wfd:OperationalCatchment[0..]
uk-wfd:inRiverBasinDistrict : uk-wfd:RiverBasinDistrict[0..]
uk-wfd:protectedArea : uk-wfd:ProtectedArea
uk-wfd:representativePoint : geometry:Point[1..1]
uk-wfd:waterBodyNotation : WaterBodyNotation
uk-wfd:HydromorphologicalDesignation
uk-wfd:GroundWaterBody
uk-wfd:groundWaterClassification : uk-wfd:GroundWaterClassification[0..]
uk-wfd:SurfaceWaterBody
uk-wfd:hydromorphologicalDesignation : uk-wfd:HydromorphologicalDesignation[1..1]
uk-wfd:immediateDownstream : uk-wfd:SurfaceWaterBody[0..]
uk-wfd:immediateUpstream : uk-wfd:SurfaceWaterBody[0..]
uk-wfd:heavilyModified : boolean[0..1]
uk-wfd:TidalRangeClassification
uk-wfd:GroundWaterClassification
uk-wfd:WaterBodyCatchment
uk-wfd:area : qudt:quantityKind value quantity:Area[0..]
uk-wfd:length : qudt:quantityKind value quantity:Length[
uk-wfd:RiverOrLake
uk-wfd:large : boolean[1..1]
uk-wfd:main : boolean[1..1]
uk-wfd:TransitionalWater
uk-wfd:salinityClassification : uk-wfd:SalinityClassification[0..]
uk-wfd:tidalRangeClassification : uk-wfd:TidalRangeClassification[0..]
uk-wfd:CoastalWater
uk-wfd:exposureClassification : uk-wfd:ExposureClassification[0..]
uk-wfd:tidalRangeClassification : uk-wfd:TidalRangeClassification[0..]
uk-wfd:lagoon : boolean
uk-wfd:CatchmentSizeClassification
uk-wfd:River
uk-wfd:altitudeClassification : uk-wfd:AltitudeClassification[0..]
uk-wfd:areaGeologyClassification : uk-wfd:AreaGeologyClassification[0..]
uk-wfd:catchmentSizeClassification : uk-wfd:CatchmentSizeClassification[0..]
uk-wfd:length : qudt:quantityKind value quantity:Length[1..1]
uk-wfd:waterBodyCatchment : uk-wfd:WaterBodyCatchment[1..1]
uk-wfd:Lake
uk-wfd:alkalinityClassification : uk-wfd:AlkalinityClassification[0..]
uk-wfd:lakeDepthClassification : uk-wfd:LakeDepthClassification[0..]
uk-wfd:ExposureClassification
uk-wfd:AreaGeologyClassification
uk-wfd:AlkalinityClassification
uk-wfd:AltitudeClassification
uk-wfd:Canal
uk-wfd:SSSIDitch
uk-wfd:SurfaceWaterTransfer
uk-wfd:LakeDepthClassification

UK Water body classifications: CatchmentSize, AreaGeology, Altitude, LakeDepth, Alkalinity, Exposure, TidalRange and HydromorpholigicalDesignation.

In addtion River water bodies may have an associated WaterBody catchment which, at least anecodally, reflects the area over which rainfall flows into a water body. Surface Water bodies can also be linked to their upstream and downstream neighbours (although currently not defined this way, *immediateUpstream* and *immediateDownstream* could be defined as sub-properties of *relatedZone*.

The local schema/ontology imports the related INSPIRE structure and as can be seen the local class hierarchy mirrors that of its parent.

One could argue that with the use of open-domained properties there is no particular need to create this mirrored structure. However, it would not be appropriate in general to associate the use of any additional properties with the classes in INSPIRE schema, also there is some appeal to containing the vocabulary within a common namespace. When vocabularies are drawn from multiple namespaces it can be confusing to know which terms come from which namespace/ontology.


wfd:WaterBody
wfd:representativePoint : geometry:Point[1..1]
zone:zoneType : zone:ZoneTypeCode
wfd:GroundWaterBody
wfd:SurfaceWaterBody
wfd:heavilyModified : boolean[0..1]
wfd:CoastalWater
wfd:RiverOrLake
wfd:large : boolean[1..1]
wfd:main : boolean[1..1]
wfd:Lake
wfd:River
wfd:TransitionalWater
wfd:Canal
wfd:SSSIDitch
wfd:SurfaceWaterTransfer
uk-wfd:WaterBody
geometry:envelope : geometry:Envelope[1..1]
uk-wfd:immediateDownstream : uk-wfd:SurfaceWaterBody[0..]
uk-wfd:immediateUpstream : uk-wfd:SurfaceWaterBody[0..]
uk-wfd:inCatchment : uk-wfd:Catchment[0..]
uk-wfd:inManagementCatchment : uk-wfd:ManagementCatchment[0..]
uk-wfd:inOperationalCatchment : uk-wfd:OperationalCatchment[0..]
uk-wfd:inRiverBasinDistrict : uk-wfd:RiverBasinDistrict[0..]
uk-wfd:protectedArea : uk-wfd:ProtectedArea
uk-wfd:representativePoint : geometry:Point[1..1]
uk-wfd:waterBodyNotation : WaterBodyNotation
uk-wfd:GroundWaterBody
uk-wfd:groundWaterClassification : uk-wfd:GroundWaterClassification[0..]
uk-wfd:SurfaceWaterBody
uk-wfd:hydromorphologicalDesignation : uk-wfd:HydromorphologicalDesignation[1..1]
uk-wfd:immediateDownstream : uk-wfd:SurfaceWaterBody[0..]
uk-wfd:immediateUpstream : uk-wfd:SurfaceWaterBody[0..]
uk-wfd:heavilyModified : boolean[0..1]
uk-wfd:CoastalWater
uk-wfd:exposureClassification : uk-wfd:ExposureClassification[0..]
uk-wfd:tidalRangeClassification : uk-wfd:TidalRangeClassification[0..]
uk-wfd:lagoon : boolean
uk-wfd:RiverOrLake
uk-wfd:large : boolean[1..1]
uk-wfd:main : boolean[1..1]
uk-wfd:TransitionalWater
uk-wfd:salinityClassification : uk-wfd:SalinityClassification[0..]
uk-wfd:tidalRangeClassification : uk-wfd:TidalRangeClassification[0..]
uk-wfd:Lake
uk-wfd:alkalinityClassification : uk-wfd:AlkalinityClassification[0..]
uk-wfd:lakeDepthClassification : uk-wfd:LakeDepthClassification[0..]
uk-wfd:River
uk-wfd:altitudeClassification : uk-wfd:AltitudeClassification[0..]
uk-wfd:areaGeologyClassification : uk-wfd:AreaGeologyClassification[0..]
uk-wfd:catchmentSizeClassification : uk-wfd:CatchmentSizeClassification[0..]
uk-wfd:length : qudt:quantityKind value quantity:Length[1..1]
uk-wfd:waterBodyCatchment : uk-wfd:WaterBodyCatchment[1..1]
uk-wfd:Canal
uk-wfd:SSSIDitch
uk-wfd:SurfaceWaterTransfer

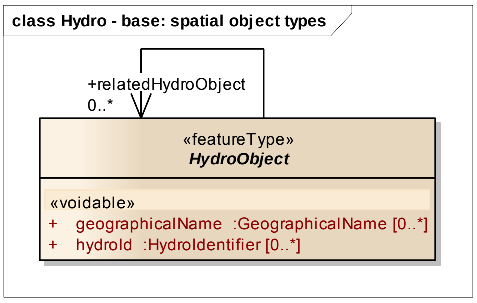
5.5 Hydrography

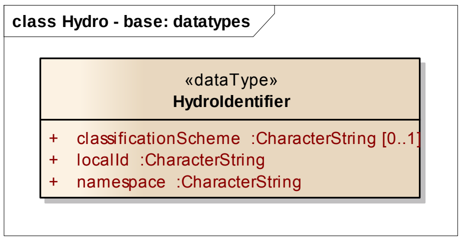
In this section we illustrate the derivation of linked data vocabuaries from the Hydrography Theme. To fulfil this task required the transformation of fragments application schema from Geographical Names,see section 5.6, and the GCM Network base model, see section 5.2.5.

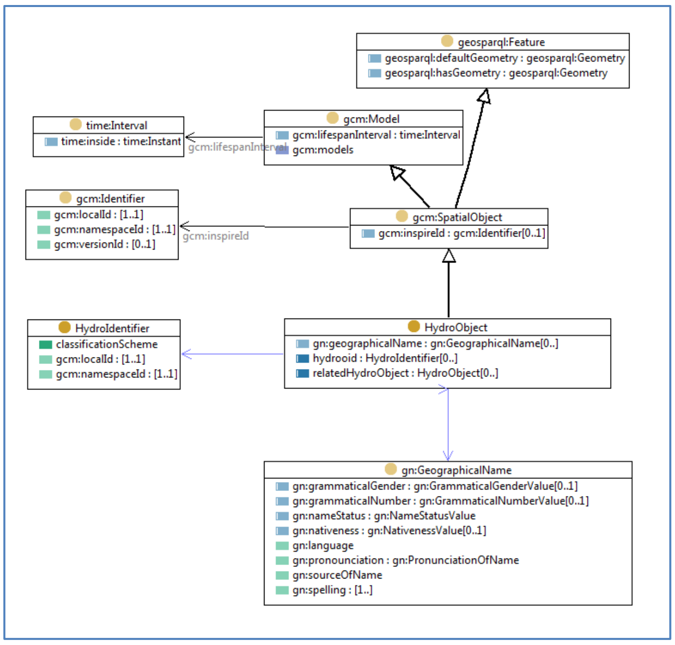
5.5.1 Hydrography Base

The following diagrams illustrate the derivation of the he ontology http://location.data.gov.uk/inspire/hy/def/hydro-base/ from the Hydro-Base application schema.

hydro-base.ttl


class Hydro - base: spatial object types
+relatedHydroObject
0..*
«featureType»
HydroObject
«voidable»
+ geographicalName :GeographicalName [0..*]
+ hydroId :HydroIdentifier [0..*]

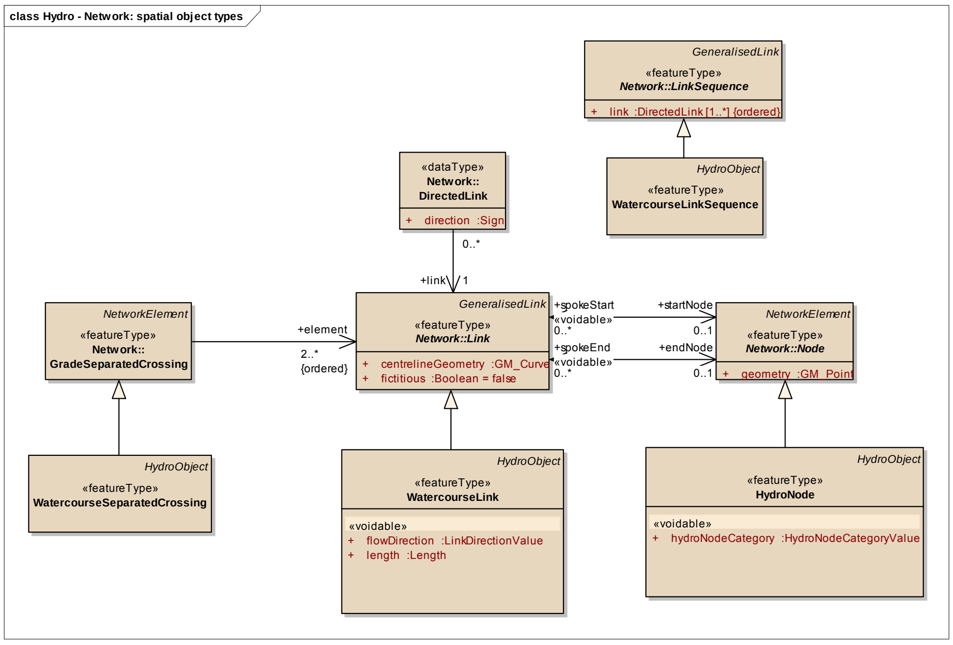

class Hydro - base: datatypes
«dataType»
HydroIdentifier
+ classificationScheme :CharacterString [0..1]
+ localId :CharacterString
+ namespace :CharacterString

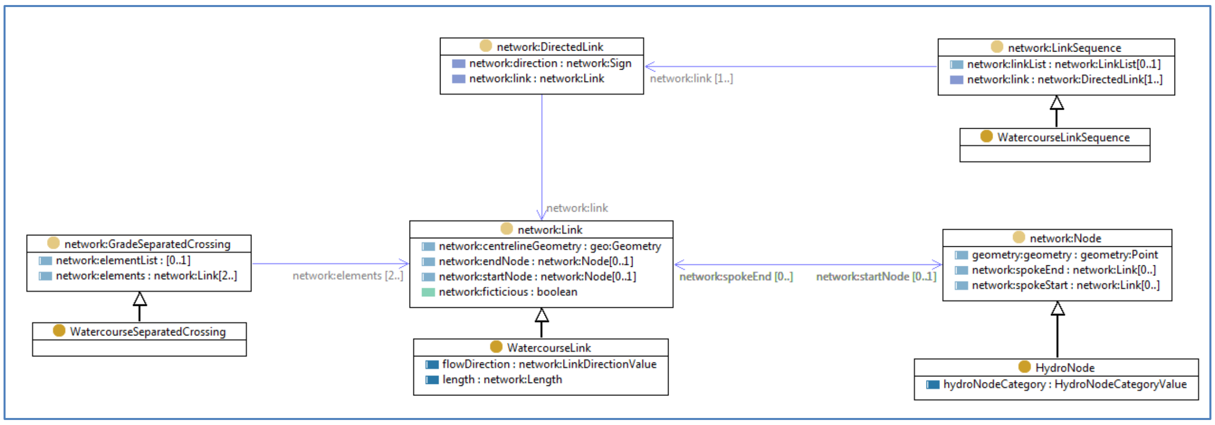

geosparql:Feature
geosparql:defaultGeometry : geosparql:Geometry
geosparql:hasGeometry : geosparql:Geometry
time:Interval
time:inside : time:Instant
gcm:lifespanInterval
gcm:Model
gcm:lifespanInterval : time:Interval
gcm:models
gcm:Identifier
gcm:localId : [1..1]
gcm:namespaceId : [1..1]
gcm:versionId : [0..1]
gcm:inspireId
gcm:SpatialObject
gcm:inspireId : gcm:Identifier[0..1]
HydroIdentifier
classificationScheme
gcm:localId : [1..1]
gcm:namespaceId : [1..1]
HydroObject
gn:geographicalName : gn:GeographicalName[0..]
hydrooid : HydroIdentifier[0..]
relatedHydroObject : HydroObject[0..]
gn:GeographicalName
gn:grammaticalGender : gn:GrammaticalGenderValue[0..1]
gn:grammaticalNumber : gn:GrammaticalNumberValue[0..1]
gn:nameStatus : gn:NameStatusValue
gn:nativeness : gn:NativenessValue[0..1]
gn:language
gn:pronounciation : gn:PronunciationOfName
gn:sourceOfName
gn:spelling : [1..]

5.5.2 Hydrography Network

The following diagrams illustrate the derivation of the he ontology http://location.data.gov.uk/inspire/hy/def/hydro-network/ from the Hydro-Network application schema.

hydro-network.ttl



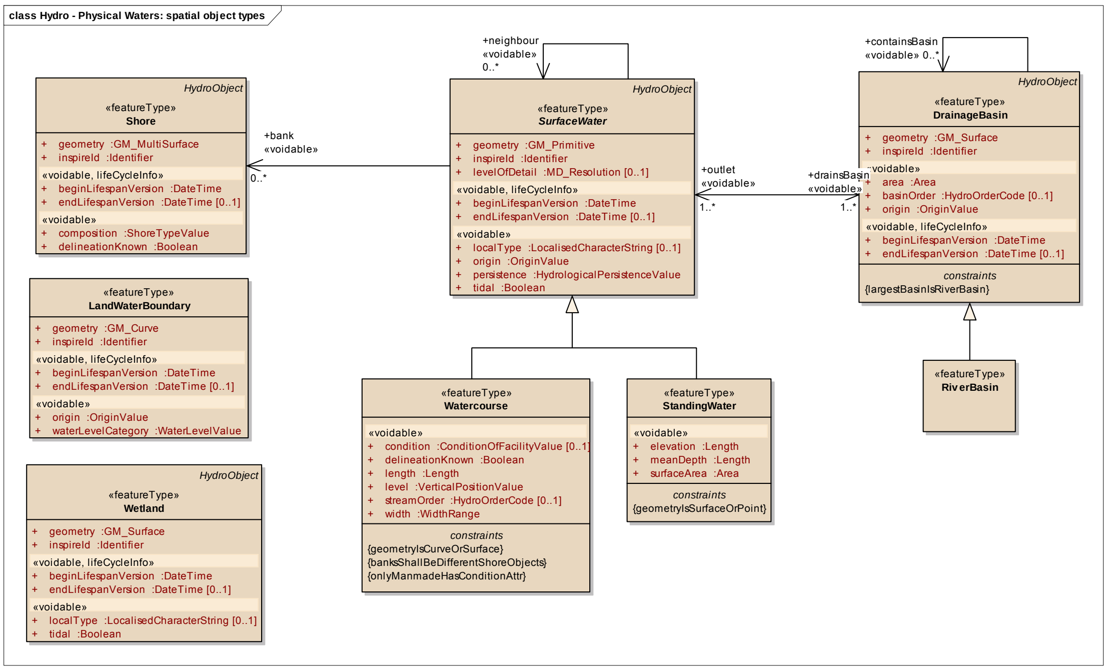

network:DirectedLink
network:direction : network:Sign
network:link : network:Link
network:link [1..]
network:LinkSequence
network:linkList : network:LinkList[0..1]
network:link : network:DirectedLink[1..]
WatercourseLinkSequence
network:link
network:GradeSeparatedCrossing
network:elementList : [0..1]
network:elements : network:Link[2..]
network:elements [2..]
network:Link
network:centrelineGeometry : geo:Geometry
network:endNode : network:Node[0..1]
network:startNode : network:Node[0..1]
network:ficticious : boolean
network:spokeEnd [0..]
network:startNode [0..1]
network:Node
geometry:geometry : geometry:Point
network:spokeEnd : network:Link[0..]
network:spokeStart : network:Link[0..]
WatercourseSeparatedCrossing
WatercourseLink
flowDirection : network:LinkDirectionValue
length : network:Length
HydroNode
hydroNodeCategory : HydroNodeCategoryValue

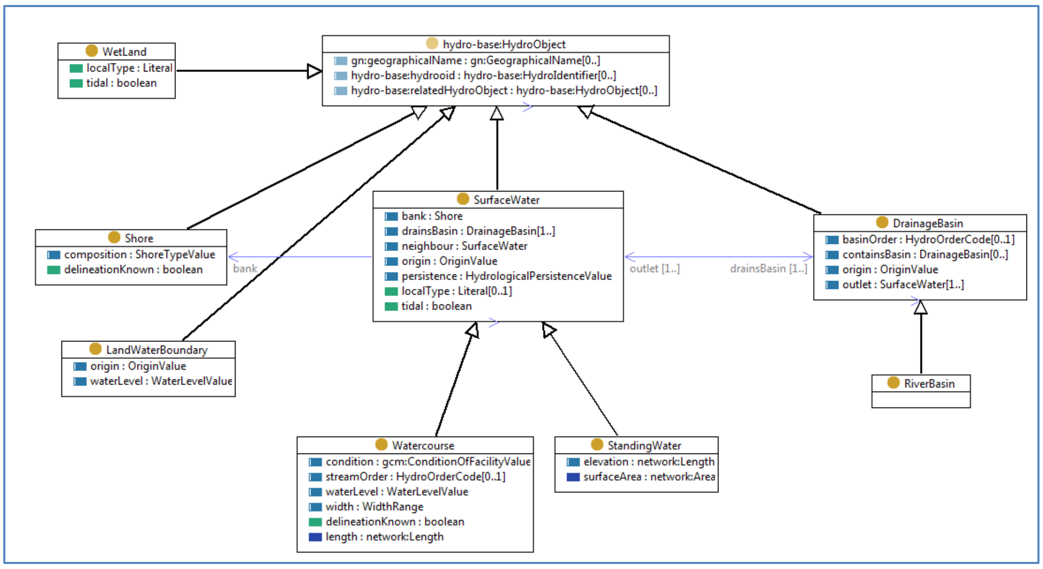
5.5.3 Hydrography Physical Waters

The following diagrams illustrate the derivation of the he ontology http://location.data.gov.uk/inspire/hy/def/hydro-physical-waters/ from the Hydro-Physical Waters application schema.

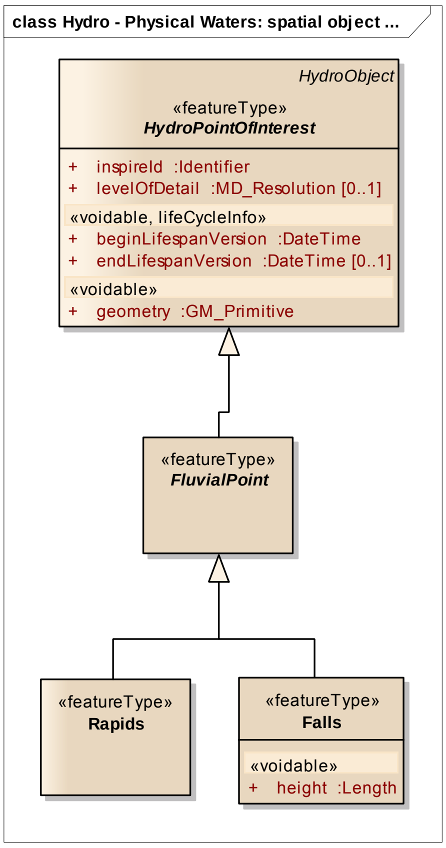
hydro-physical-waters.ttl

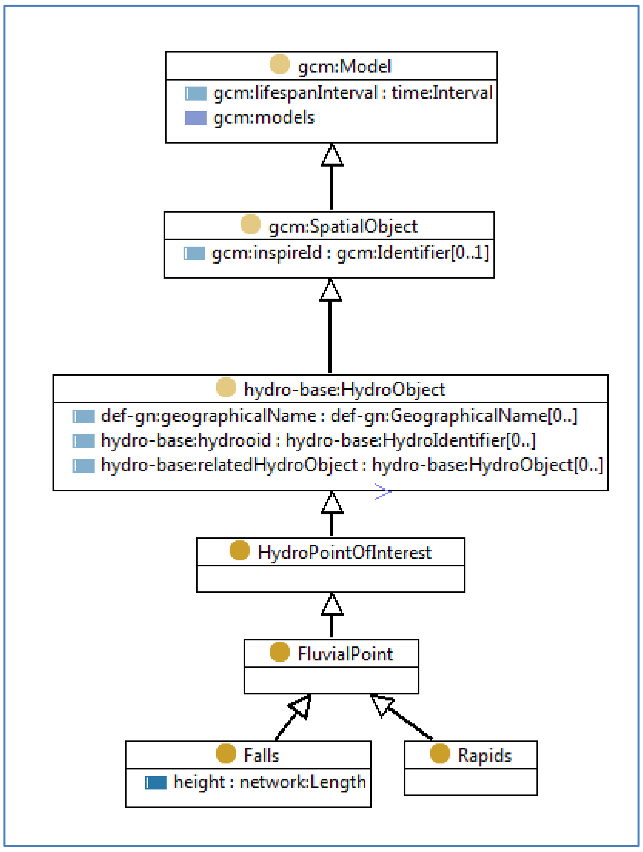
Physical Waters Spatial Objects


class Hydro - Physical Waters: spatial object types
HydroObject
«featureType»
Shore
+ geometry :GM_MultiSurface
+ inspireId :Identifier
«voidable, lifeCycleInfo»
+ beginLifespanVersion :DateTime
+ endLifespanVersion :DateTime [0..1]
«voidable»
+ composition :ShoreTypeValue
+ delineationKnown :Boolean
+bank
«voidable»
0..*
+neighbour
«voidable»
0..*
HydroObject
«featureType»
SurfaceWater
+ geometry :GM_Primitive
+ inspireId :Identifier
+ levelOfDetail :MD_Resolution [0..1]
«voidable, lifeCycleInfo»
+ beginLifespanVersion :DateTime
+ endLifespanVersion :DateTime [0..1]
«voidable»
+ localType :LocalisedCharacterString [0..1]
+ origin :OriginValue
+ persistence :HydrologicalPersistenceValue
+ tidal :Boolean
+outlet
«voidable»
1..*
+drainsBasin
«voidable»
1..*
+containsBasin
«voidable» 0..*
HydroObject
«featureType»
DrainageBasin
+ geometry :GM_Surface
+ inspireId :Identifier
«voidable»
+ area :Area
+ basinOrder :HydroOrderCode [0..1]
+ origin :OriginValue
«voidable, lifeCycleInfo»
+ beginLifespanVersion :DateTime
+ endLifespanVersion :DateTime [0..1]
constraints
{largestBasinIsRiverBasin}
«featureType»
LandWaterBoundary
+ geometry :GM_Curve
+ inspireId :Identifier
«voidable, lifeCycleInfo»
+ beginLifespanVersion :DateTime
+ endLifespanVersion :DateTime [0..1]
«voidable»
+ origin :OriginValue
+ waterLevelCategory :WaterLevelValue
«featureType»
Watercourse
«voidable»
+ condition :ConditionOfFacilityValue [0..1]
+ delineationKnown :Boolean
+ length :Length
+ level :VerticalPositionValue
+ streamOrder :HydroOrderCode [0..1]
+ width :WidthRange
constraints
{geometryIsCurveOrSurface}
{banksShallBeDifferentShoreObjects}
{onlyManmadeHasConditionAttr}
«featureType»
StandingWater
«voidable»
+ elevation :Length
+ meanDepth :Length
+ surfaceArea :Area
constraints
{geometryIsSurfaceOrPoint}
«featureType»
RiverBasin
HydroObject
«featureType»
Wetland
+ geometry :GM_Surface
+ inspireId :Identifier
«voidable, lifeCycleInfo»
+ beginLifespanVersion :DateTime
+ endLifespanVersion :DateTime [0..1]
«voidable»
+ localType :LocalisedCharacterString [0..1]
+ tidal :Boolean

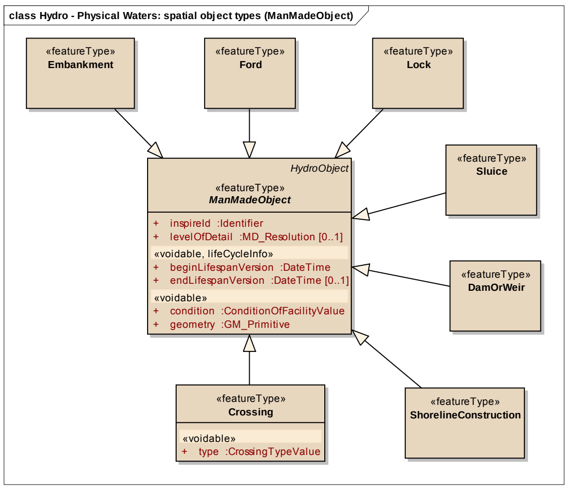

WetLand
localType : Literal
tidal : boolean
hydro-base:HydroObject
gn:geographicalName : gn:GeographicalName[0..]
hydro-base:hydrooid : hydro-base:HydroIdentifier[0..]
hydro-base:relatedHydroObject : hydro-base:HydroObject[0..]
Shore
composition : ShoreTypeValue
delineationKnown : boolean
bank
SurfaceWater
bank : Shore
drainsBasin : DrainageBasin[1..]
neighbour : SurfaceWater
origin : OriginValue
persistence : HydrologicalPersistenceValue
localType : Literal[0..1]
tidal : boolean
outlet [1..]
drainsBasin [1..]
DrainageBasin
basinOrder : HydroOrderCode[0..1]
containsBasin : DrainageBasin[0..]
origin : OriginValue
outlet : SurfaceWater[1..]
LandWaterBoundary
origin : OriginValue
waterLevel : WaterLevelValue
RiverBasin
Watercourse
condition : gcm:ConditionOfFacilityValue
streamOrder : HydroOrderCode[0..1]
waterLevel : WaterLevelValue
width : WidthRange
delineationKnown : boolean
length : network:Length
StandingWater
elevation : network:Length
surfaceArea : network:Area

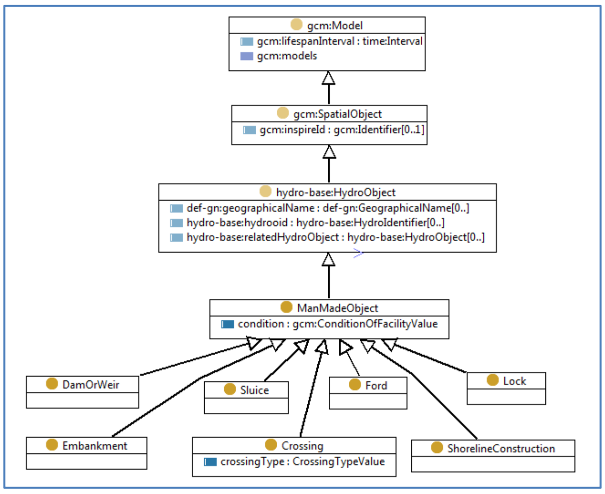
Physical Waters Spatial Objects -Hydro Points of Interest




gcm:Model
gcm:lifespanInterval : time:Interval
gcm:models
gcm:SpatialObject
gcm:inspireId : gcm:Identifier[0..1]
hydro-base:HydroObject
def-gn:geographicalName : def-gn:GeographicalName[0..]
hydro-base:hydrooid : hydro-base:HydroIdentifier[0..]
hydro-base:relatedHydroObject : hydro-base:HydroObject[0..]
HydroPointOfInterest
FluvialPoint
Falls
height : network:Length
Rapids

Physical Waters Spatial Objects -Man Made Objects

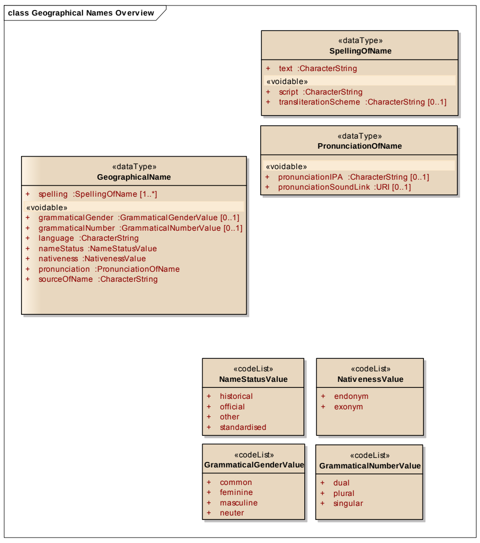

class Hydro - Physical Waters: spatial object types (ManMadeObject)
«featureType» Embankment
«featureType» Ford
«featureType» Lock
HydroObject
«featureType» ManMadeObject
+ inspireId :Identifier
+ levelOfDetail :MD_Resolution [0..1]
«voidable, lifeCycleInfo»
+ beginLifespanVersion :DateTime
+ endLifespanVersion :DateTime [0..1]
«voidable»
+ condition :ConditionOfFacilityValue
+ geometry :GM_Primitive
«featureType» Sluice
«featureType» DamOrWeir
«featureType» Crossing
«voidable»
+ type :CrossingTypeValue
«featureType» ShorelineConstruction

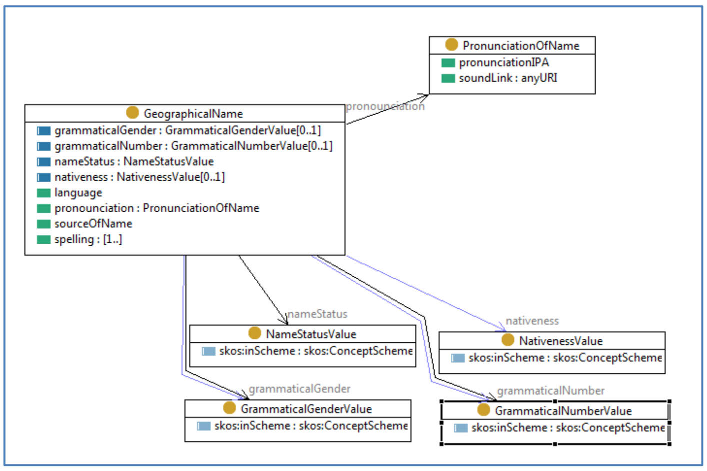

gcm:Model
gcm:lifespanInterval : time:Interval
gcm:models
gcm:SpatialObject
gcm:inspireId : gcm:Identifier[0..1]
hydro-base:HydroObject
def-gn:geographicalName : def-gn:GeographicalName[0..]
hydro-base:hydrooid : hydro-base:HydroIdentifier[0..]
hydro-base:relatedHydroObject : hydro-base:HydroObject[0..]
ManMadeObject
condition : gcm:ConditionOfFacilityValue
DamOrWeir
Sluice
Ford
Lock
Embankment
Crossing
crossingType : CrossingTypeValue
ShorelineConstruction

5.6 Geographical Names

The following diagrams illustrate the derivation of the ontology http://location.data.gov.uk/inspire/gn/def/geographical-names/ from the Geographical Names application schema.

geographical-names.ttl


class Geographical Names Overview
«dataType» SpellingOfName
+ text :CharacterString
«voidable»
+ script :CharacterString
+ transliterationScheme :CharacterString [0..1]
«dataType» PronunciationOfName
«voidable»
+ pronunciationIPA :CharacterString [0..1]
+ pronunciationSoundLink :URI [0..1]
«dataType» GeographicalName
+ spelling :SpellingOfName [1..*]
«voidable»
+ grammaticalGender :GrammaticalGenderValue [0..1]
+ grammaticalNumber :GrammaticalNumberValue [0..1]
+ language :CharacterString
+ nameStatus :NameStatusValue
+ nativeness :NativenessValue
+ pronunciation :PronunciationOfName
+ sourceOfName :CharacterString
«codeList» NameStatusValue
+ historical
+ official
+ other
+ standardised
«codeList» NativenessValue
+ endonym
+ exonym
«codeList» GrammaticalGenderValue
+ common
+ feminine
+ masculine
+ neuter
«codeList» GrammaticalNumberValue
+ dual
+ plural
+ singular


PronunciationOfName
pronunciationIPA
soundLink : anyURI
pronounciation
GeographicalName
grammaticalGender : GrammaticalGenderValue[0..1]
grammaticalNumber : GrammaticalNumberValue[0..1]
nameStatus : NameStatusValue
nativeness : NativenessValue[0..1]
language
pronounciation : PronunciationOfName
sourceOfName
spelling : [1..]
nameStatus
NameStatusValue
skos:inScheme : skos:ConceptScheme
nativeness
NativenessValue
skos:inScheme : skos:ConceptScheme
grammaticalGender
GrammaticalGenderValue
skos:inScheme : skos:ConceptScheme
grammaticalNumber
GrammaticalNumberValue
skos:inScheme : skos:ConceptScheme

5.7 Adapting for Objects as Graphs

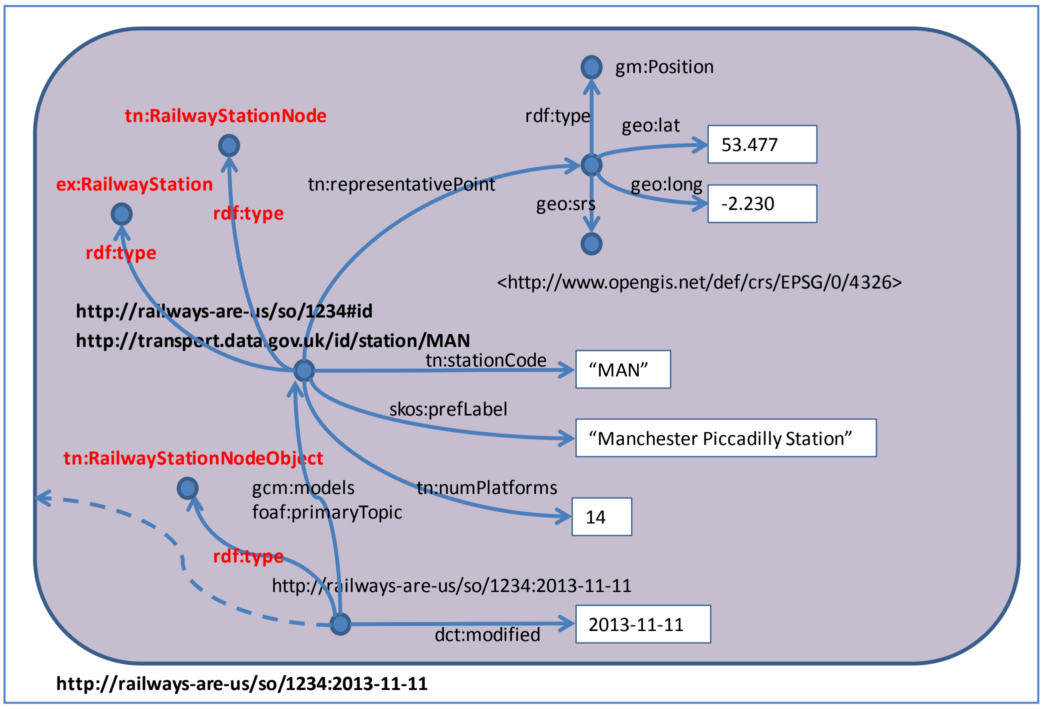
[This section added post workshop meeting 15th April 2014]

The vocabularies presented throughout this section follow the 'objects as node' approach (see 3.2.2). Since the concenus of our face-to-face meeting was either to focus on the objects as graphs approach (see 3.2.1) or include both approaches in 'the experiment' this section has been added to discuss briefly how such a change would affect the models/vocabularies outlined in this section (and attached as embedded objects). Two possible variations come to mind:

- Use the derived spatial-object classes (eg. *tn:RailwayStationNode*) as 'mix-in' classes attributing *rdf:type* to the abstracted-thing. This allows more narrowly defined property domains and ranges to be expressed with are property domain/range or as class specific restrictions as has been done through the vocabularies given here. A single generic spatial-object class that can be used as domain or range of properties that are 'about' the spatial-object (as a graph) itself rather than the 'thing' that the object abstracts. It may be useful to create two classes per spatial-object type, one as a 'mix-in' applied to the abstracted 'thing' and to serve in the expression of property domain/range constraints on properties that are 'about' the abstracted thing and a second to serve as 'tag' on the spatial object that is indicative of the abstraction contained there-in. The information associated with these second classes could go further and enumerate the properties that can be used to described the abstracted 'thing' as illustrated below:

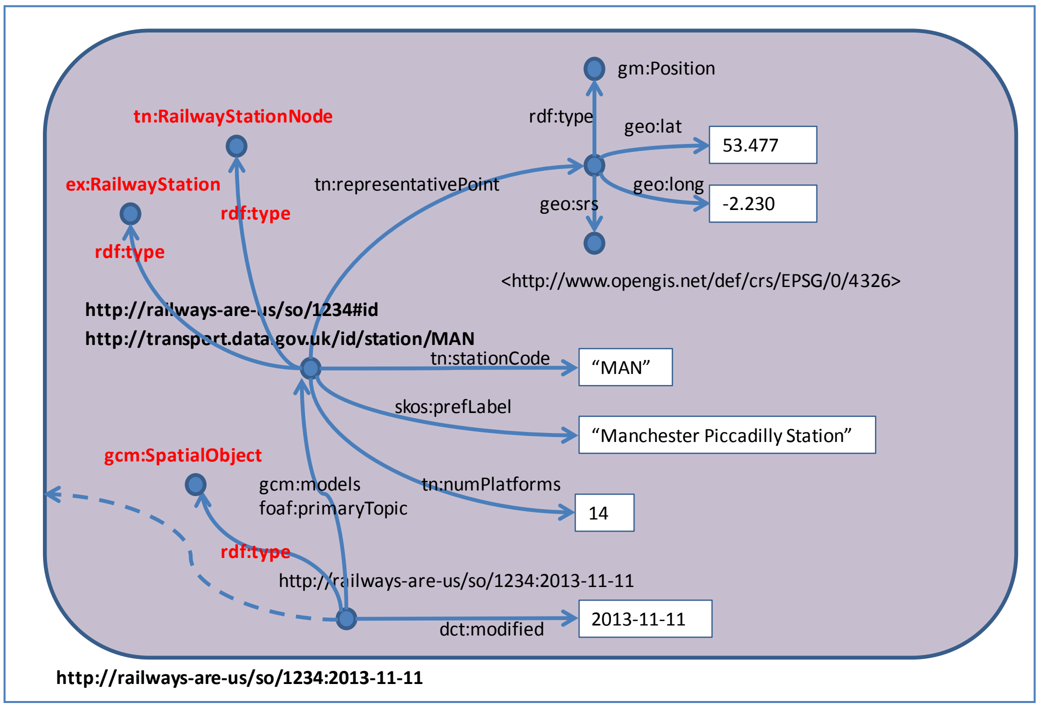
```
tn:RailwayStationNode
        a       owl:Class ;
        owl:disjointWith       tn:RailwayStationNodeObject ;
        rdfs:label     "RailwayStationNode"@en;
        rdfs:comment   """A mix-in class applied to entities that in some cases may be
regarded as nodes in a railway transport network - typically, but not exclusively,
railway stations."""@en ;
        # Restriction to indicate expected use of stationCode with RailwayStationNode
        rdfs:subClassOf [
                a       owl:Restriction
                owl:onProperty         tn:stationCode ;
                owl:minCardinality     0
        ]
        ##... more restrictions to indicate use of other properties.
        .

tn:RailwayStationNodeObject
        owl:disjontWith        tn:RailwayStationNode ;
        rdfs:label             "RailwayStationNodeObject"@en;
        rdfs:comment           """A class for spatial-object graphs expected to
describe 'thing' using only the properties associated with describing a
RailwayStationNode.""" ;
        rdfs:subClassOf               gcm:SpatialObject ;
        ## Loose restriction to RailwayStationNode expressions
        rdfs:subClassOf [
                a       owl:Restriction ;
                owl:onProperty         gcm:models
                owl:allValuesFrom      tn:RailwayStationNode
        ] ;
        ## Annotation properties to enumerate allowed and voidable properties.
        gcm:allowedProperties          (tn:stationCode tn:numPlatforms ...) ;
        gcm:voidableProperties         (....) ;
        .
```

To avoid entering owl full *gcm:allowedProperties* and *gcm:voidableProperties* would need to be defined as OWL annotation properties (and thus would have no semantic consequences within OWL). The two series of classes, for abstracted things and for spatial-objects are mutually disjoint ie. *tn:RailwayStationNode* and *tn:RailwayStationNodeObject* are disjoint. Instance data for our running railway station example is illustrated below:


gm:Position
rdf:type
geo:lat
53.477
tn:RailwayStationNode
ex:RailwayStation
tn:representativePoint
geo:long
-2.230
rdf:type
geo:srs
rdf:type
<http://www.opengis.net/def/crs/EPSG/0/4326>
http://railways-are-us/so/1234#id
http://transport.data.gov.uk/id/station/MAN
tn:stationCode
"MAN"
skos:prefLabel
"Manchester Piccadilly Station"
tn:RailwayStationNodeObject
gcm:models
foaf:primaryTopic
tn:numPlatforms
14
rdf:type
http://railways-are-us/so/1234:2013-11-11
dct:modified
2013-11-11
http://railways-are-us/so/1234:2013-11-11

The core principle here is that one has distinct classes for abstracted 'thing' and spatial-object. One could equally choose to retain *tn:RailwayStationNode* to type the spatial-object (as a graph) and create a second distinguished class for the abstracted 'thing', e.g. *tn:RailwayStationNodeThing* which can be in the expression of property domain, range and cardinality restrictions.

- Alternatively, eliminate all domain and range constraints, both expressed using *rdfs:domain* and *rdfs:range* and as *owl:onProperty* class specific restrictions. Common properties about spatial-objects (graphs) can have their domains restricted to a common *gcm:SpatialObject* class which itself is disjoint from all derived 'mix-in' classes. This disjointness is best accomplished with a top-level class disjoint with *gcm:SpatialObject* and from which the derived 'mix-in' classes are subclassed (making them necessarily disjoint with *gcm:SpatialObject*). Very little difference is evident in the resulting RDF, illustrated below, only that a single *gcm:SpatialObject* class is used in place of more distinguished spatial-object subclasses.



The first of these approaches retains much/all of the formulation presented earlier in this section, except that at the very top-level, the derived mix-in classes are **not** subclassed from a common *gcm:SpatialObject* class - indeed they are necessarily disjoint from it. High level properties associated with the spatial-object itself, sometimes thought of as object metadata, are expressed in RDF triples with the object itself as subject whilst statements using properties associated with the 'thing' abstracted by the 'object' use a distinct subject URI that designates that 'thing'rather than the spatial-object.

The second of these approaches leaves the resulting RDF vocabularies very free and without much semblance of constraint expressed in the application schema from which the vocabulary is derived. This may leave many feeling uncomfortable. One can progress incrementally back to the first of these approaches through the inclusion of domain, range and cardinality constraints on property use with respect to derived 'mix-in' classes, and if desired, distinguished subclasses of *gcm:SpatialObject* corresponding to each mix-in class which call also serve to catalogue the permissible and voidable properties of the spatial object.

6 Concluding Remarks

The transformation of INSPIRE application schema into RDF/OWL ontologies is largely tractable, however there are some issue of detail that need to be considered:

1. It is not clear whether and how to reflect the use of stereotypes in UML into the OWL
2. Versioning requirements (single current version with/with out versionIds or with 'history') impact the conceptualisation of spatial objects - eg. does and object identifier identify and object (over all its lifetime), an object version (snapshot), the container of the current state or some combination of all of the above. These affect how object references are made using URI and how instance data is transformed into RDF.
3. The generation of instance data is significantly affected by:
   - when making assertions about a real-work 'thing' RDF subject URI designate that real-world 'thing' or an 'object' that models or abstracts it.

    - whether spatial-object types can be used as mix-ins or not. ie. whether a spatial object can be both say a *RailwayStationNode* and a *RailwayStationArea* or not.
    - whether derived types pertain to the abstraction (a spatial-object) or the abstracted 'thing' (e.g. a railway station) or whether there is a distinct series for both.

4. There are a number of recurring patterns in the UML models that it would be good to factor out so that they can be implemented once rather than 're-invented' in successive application schema. This also has a bearing on property reuse. An RDF ideal is to leave properties available for re-use by giving them open-domains. However, one needs to be able to 'spot' properties whose purpose/role in relation some subject class is sufficiently the same as another property used with another class to motivate the use of one common property.
5. There needs to be some common practice with respect to geometry representation in RDF.
6. There are vocabulary management issues that we haven't tackled in this document. Versioning vocabularies; the factoring of codelists and enumerations.
7. In transforming there GCM Network application schema there is a requirement for order associations with cardinality 2 or more items. RDF collections (*rdf:Seq*) could be used, but are largely deprecated. The only commonly used ordered structure in RDF is *rdf:List*. In these cases we used a cardinality constraint on a multi-valued property and an *rdf:List* to provide ordering with the expectation that the multi-valued property would enumerate the members of the list. There is probably a more elegant way to accomplish this.

7 Annex 1 Derived Vocabulary Bundle

The attached file contains a bundle of INSPIRE related derived ontologies discussed in this report arising from two projects with the UK Environment Agency and additional ontologies created for this report - principally the Hydrography application schema. Note these were prepared using an "objects as nodes" approach (see 3.2.2)

derived-vocabulary-bundle.zip