Epimorphics Ltd
Linked data solutions

# INPSIRE in RDF

Deliverable 2

Version 0.2
19th May 2014

## Changelog

| Version | Date | Editor | Notes |
|---|---|---|---|
| 0.1 | 1/05/2014 | Stuart Williams | Initial Draft |
| 0.2 | 19/05/2014 | Stuart Williams | Added intro and section 2 |

# INPSIRE in RDF

Deliverable 2

## Contents


Changelog ..... ii
Contents ..... iii
1 Introduction ..... 1
2 Common Approach ..... 1
2.1 Conceptual Issues ..... 1
2.1.1 Spatial Object formulation in RDF ..... 1
2.1.2 Spatial Object Versioning ..... 2
2.2 Schema Transformation Issues ..... 3
2.2.1 Property Reuse ..... 3
2.2.2 Making use of common Linked Data vocabularies ..... 3
2.2.3 Cardinality Constraints ..... 3
2.2.4 Voidable properties ..... 3
3 Schema and Data Conversion Tools ..... 3
3.1 Schema Conversion ..... 3
3.2 Instance Data Conversion ..... 4
3.2.1 Conversion Tools ..... 6
3.2.1.1 DCLIB ..... 6
3.2.1.2 OpenRefine (formerly Google Refine, formerly GridWorks)) ..... 7
3.2.1.3 Topbraid Composer ..... 8
3.2.2 Commercial ETL Tools ..... 9
4 Architecture and Infrastructure Components ..... 9
4.1 Registers and Registries ..... 9
4.2 Metadata and Catalogues ..... 11
4.3 Geospatial Services ..... 13

# 1 Introduction

This report is the second deliverable due as an expert contribution to the Are[3]na investigation into INSPIRE in RDF. It contains discussion of:

- issues that remain with respect to an emerging common approach to the transformation of INSPIRE application schema into RDFS/OWL vocabularies (see section 2)
- schema transformation tools (see section 3.1)
- instance data conversion tools/approaches (see section 3.2)
- architecture and infrastructure components (see section 4)

# 2 Common Approach

The proposed common approach is based largely around the application of DIS-19150-2 transformation rules, with some variation which themselves may be influential on the onward development of that specification toward publication as an international standard.

## 2.1 Conceptual Issues

### 2.1.1 Spatial Object formulation in RDF

Whilst there are some issues of detail - e.g. the degree to which UML schema, package and class names are embedded into the URI for derived properties, and the rules for reusing similarly (local) named and spirited properties rather than creating multiple properties with substantially the same definition. The most significant issue remains settling the way in which instance data is to be formulate. In our earlier deliverable we presented three different formulations of spatial-objects (feature instances) in RDF:

| Spatial-Objects As... | Description | Pros | Cons |
|---|---|---|---|
| Nodes | Spatial-Object as the principal subject.<br><br>Where possible make links to the real-world phenomenon using gcm:models | Can make natural use of RDFS/OWL to express the derived data model, e.g. to restrict the use of properties to particular spatial-object types.<br><br>Can easily separate RDF statements from different spatial-objects on the basis of statement subject.<br><br>Doesn't require real-world subject URI, but can usefully link to them where available (gcm:models). Similar in form to thematic references.<br><br>Easily deployed in a single default graph triplestore or as materialised web documents (no query capability in latter case) | Indirect statements:<br>Whilst the spatial-object is the explicit subject of the RDF statements the real subject of what is being said is the abstracted real-world phenomenon.<br><br>Risks conflating abstracted 'thing' and abstracting object. |

| Spatial-Objects As... | Description | Pros | Cons |
|---|---|---|---|
| Graphs | Spatial-objects as a URI named collection of statements (a graph) about some real-world phenomenon plus statements about the spatial-object(graph) itself. | Use's subject URI that separately designate real-world thing and abstracting spatial-object/graph.<br><br>Speaks clearly, separately and directly about spatial object and abstracted 'thing' i.e. addresses conflation issue.<br><br>Easily deployed as materialised web documents (no query capability in this case).<br><br>Uses feature-types as mixin classes which 'attract' property usage on real-world subjects. | Cannot use RDFS/OWL to control statement usage within the spatial-object graph (but in general you can't anyway due to OWA).<br><br>Relies on graph containment to separate statements contributed by different spatial-objects, so cannot be effectively deployed in a single default graph. Requires quad-store for effective queriable depolyment.<br><br>Quadstores queries are more complex. |
| Nodes and Graphs | As 'Nodes' above but wrapped in a graph/document to provide an anchor for object metadata.<br><br>There are now three subjects: a real-world thing; a spatial-object that models it; and a graph/document which describes both itself and the spatial object. | Can make natural use of RDFS/OWL to express the derived data model, e.g. to restrict the use of properties to particular spatial-object types.<br><br>Doesn't require real-world subject URI, but can usefully link to them where available (gcm:models). Similar in form to thematic references.<br><br>Addresses conflation issue.<br><br>With care can be deployed in a single default graph because spatial-object and document subject can be used to isolated statements arising from a single spatial object. | Indirect statements:<br><br>1) statements made using the spatial-object as subject are indirectly really about the real-world thing;<br><br>2) statements made using the graph/document as subject are indirectly really about the spatial-object. |

The choice amongst the alternate formulations has its most profound affect on the generate RDF instance data. It has a much less marked affect on the translation of INSPIRE application schema into RDF vocabularies. "..as Nodes" and "..as Nodes and Graphs" are the most pragmatic and once one admits the third subject (the document describing the spatial-object) they are more or less identical. Albeit as an indirect subject, the inclusion of the document subject, but the need for graph encapsulation makes query-able deployment more complex, requiring a quad-store rather than a triple-store.

Making this choice is by far the most important open issue that needs to be settled.

### 2.1.2 Spatial Object Versioning

INSPIRE spatial-object identifiers made up of a *namespaceId*, *localId* and *versionId* are strongly suggestive of the notion that spatial-objects are versioned. If *versionId* is embedded within an spatial-object URI the expectations of URI persistence imply that that version will continue to be available on an on-going basis. Conversely, a *versionId* may simply be an attribute of spatial-object that indicates a change in the objects 'state'.

Three versioning models come to mind:

- Single current version (unversioned)
- Single current version (versioned)
- Versioned Spatial-Objects with version history

In all three cases an unversioned URI references designates the spatial-object over its entire history. Retrieval of using the unversioned URI retrieves the most recent available version of the spatial-

objects' state; or in the case of the versioned spatial-object the most recent/current state and/or a catalog of available versions.

The use of versioned spatial object URI is only relevant where spatial-objects are to versioned and the version history is maintained.

## 2.2 Schema Transformation Issues

### 2.2.1 Property Reuse

The detection of and rules for processing identically named and spirited UML association roles into shared RDF properties across UML classes within the same application schema and in different application schema needs further study and experiment.

### 2.2.2 Making use of common Linked Data vocabularies

The use/re-use of existing linked-data vocabularies such as FOAF, SKOS, DCAT, VoID, ORG, CUBE, SSN and others needs further study and experiment. Concrete rules can only be established on the basis of an assessment of the suitability of a given vocabulary to represent data required of a given INSPIRE schema. A softer approach may be to frame more general guidance about the adoption of external vocabularies is specialisation of INSPIRE scheme (national or organisational specialisations), allow practice to evolve and to consolidate 'best' practice at a later date in the light of experience. This will require that practitioners have access to tooling that enables them to specify rules for the incorporation of external vocabularies of their choosing.

### 2.2.3 Cardinality Constraints

Preserving cardinality constraints expressed in UML into derived RDFS/OWL has limited use from the point of view of validating spatial objects, because in RDFS/OWL they are not an expression of syntactic constraints and the open-world-assumption applies (statements do not have to be made - and their absence in not render a model invalid). However, their preservation does usefully convey modelling intend both to data publishers and data consumers.

### 2.2.4 Voidable properties

RDF has no natural means to express a voided property value. The absence of a property value from an RDF cannot be used to infer that the property has no value, only that the properties value is unknown or has not been disclosed.

One could conceive of a small set of void values used to convey void reasons and create unionOf property ranges that extend the 'natural' range of the voidable property with the set of available void reasons. However this is rather ugly.

An alternate approach is to annotate instances with explicit information about properties that have been given void values and the reason for the void.

It is also reasonable to regard the simple absence of a voidable property as a void without reason.

# 3 Schema and Data Conversion Tools

## 3.1 Schema Conversion

To date our process has been 'manual' based on both the narrative INSPIRE data specifications and on the diagrams available in the consolidated UML model. This is laborious and unsustainable - however it is/has been useful in exploring what an automated transformation would need to do and where elements of human supervision or special casing (schema-specific 'rules') might be required.

Options for automated schema conversion include:

- ShapeChange (http://shapechange.net) looks like a very promising tool for automating the conversion INSPIRE application schema to corresponding RDF/OWL vocabularies once the particular transformation rules are settled.

  We have had a 'quick' go at transforming a version of the INSPIRE application schema using a relatively minimal ShapeChange configuration. We've noticed a few anomalies the generated output, but nothing that could not be addressed, and many of which could be addressed with a more sophisticated configuration or might have arisen due to user error.

- Fullmoon (https://code.google.com/p/fullmoon-framework/) from CSIRO may have broadly similar capabilities to ShapeChange however it is harder to set up and experiment with. So far we have not managed to use it successful, but have only invested a small amout of time in trying.

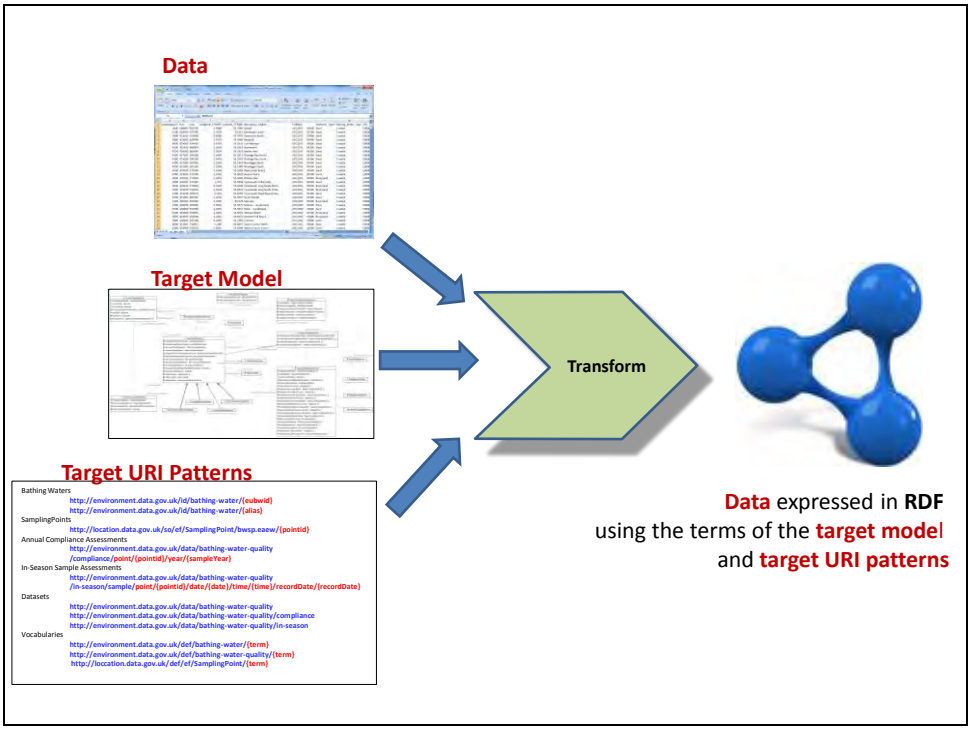
- XSLT transform from the UML/XMI files. However this requires a detailed knowledge of the XMI formats (or other XML based source formats) and much of that is already encapsulated in ShapeChange/Fullmon.

Based on limited practical experience with these tools we would recommend working with the contributors to ShapeChange to ensure that it is capable of implementing the transformation steps that are finalised for this project.

## 3.2 Instance Data Conversion

The transformation of 'instance' data into RDF expressed using a particular set of vocabularies, in this case derived from INSPIRE application schema, is quite different from the process of transforming the application schema itself and likely requires some different tools (or at least different members from a suite of tools). The tasks is illustrated in the diagram below and apart from the data that is going to be presented as or transformed into RDF, there are two other necessary pieces of input. Firstly the target linked data vocabulary/model which has been the main focus of this project; and secondly a plan for the linked data URI space where the data is to be published. This latter interacts with notions of persistent URI and 'the publishing commitment'[1]

[1] In publishing the data what is the publishers commitment to its continued availability; at the URI where it is initial published; how to date will it be maintained; and how much notice of change will be given in the event that the data is to be retired r relocated. All questions of governance.




It is also necessary think about how the publication is going to be organised and how it is going to be maintained and updated - particularly with respect to the persistence of links that others make to the published data.

The transformed data may be:

- materialised as individual *'documents'* (.json, .gml, .rdf, .ttl, .html) published conventionally on a web server. Each *'document'* is effectively a 'small' graph (a spatial-object) about some primary topic that may carry its own '*document*' metadata.

  Works for "spatial-objects as nodes" or "spatial-objects as graphs" approaches.

  This style of publication enable 'link-following', but provides no means to query the data based in its content.

- published, in aggregate, into the default-graph of triple store. This looses object boundaries in the sense that it is not possible in general to segregate statements back into groupings represent the individual *'documents'* that were contributed.

  Good for "Spatial-Objects as Nodes" approach only. Intrinsically queriable; linked data URI can be 'animated' by frontend technologies ranging from Apache mod_rewrite to convert request URI into SPARQL describe queries; to Pubby which provides RDF and HTML output formats; and LDA (linked-data api - Elda and Puelia implementations) which provides URI based querying capabilities and RDF, HTML and developer centric JSON and XML formats.

- published as named graphs in a Quad Store (with or without a UNION default graph). Very like publishing materialised *documents* except each document/spatial-object(-version) is publishing as a distinct graph in a SPARQL dataset. A UNION default graph gives a merged view with potential for different object versions and/or different objects to provide contradictory views via the default graph.

  Good for "Spatial-Objects a Graphs" and "Spatial-Objects as Nodes" approaches. Intrinsically queryable; Linked-data URI can be 'animated' as above.

These approaches are all based on some form of Extract, Transform and Load (ETL) model of publishing.

Alternatively, data maybe

- left within an RDBMS can be left in place (or in a non mission critical mirror) and a query translation data base adapter eg D2RQ can be use to create as SPARQL endpoint that maps inbound SPARQL request to SQL queries and SQL resultsets either into SPARQL result set (SPARQL select queries) or RDF graphs (SPARQL described and construct queries). Once a SPARQL endpoint is achieve, link-following and/or URI based querying can be obtained in the same way (mod_rewrite, Pubby or LDA... or something similar).

More recently the W3C have released as specification R2RML that can support the definition of both query transformation and ETL transformation of the data.

In the work we have done to date we generally use an ETL based approach, using custom transformation written in either Java, Ruby or more recently a simple data conversion templating library/language named DCLIB (more below).

Query transformation approaches involve the creation of a mapping file (say in R2RML or in D2RQ Mapping Language). Both commercial and open-source implementations of R2RML exist, though we have no experience of using them in practice.

### 3.2.1 Conversion Tools

Tooling for converting data into RDF formats is relatively ad-hoc - unlike the eco-systems that have grown up for getting data into and out of RDBMS systems.

In an ideal world something similar to the visual mapping UI of Microsoft's Sql Server Integration Services (SSIS) with escapes in to programming to complex tasks: that helps builds conversion workflows that include: data cleaning; reconciliation against and substitution of authoritative reference data as well as data transformation. However, we are not aware of such a framework targetted generation of linked data.

#### *3.2.1.1 DCLIB*

DCLIB (DataConversionLibrary) is an open-source project from Epimorphics. It provides a data conversion library that can be built into a larger data conversion workflow and a simple command line tool for the direct execution of transformation. We have build this library to support data conversion task that we undertake for clients. Below is an example of a small template that is used to convert a simple two column code list into RDF. The template is written as a JSON Object. The top-level "*Composite*" template uses the *oneOffs* array of templates to generate preamble for the output and then applies each of the *templates* to each row of the input table. Each template uses has an *@id* field which is used to generate the subject URI for the triples generated by the template. The fields that follow the *@id* field provide property/value pairs for that common subject. Values enclosed in angle brackets generate URI values (a opposed to literal values). Most field can be either single valued or array valued.

```
[ { name                : "role",
    required : ["role_code","role_desc"],
    type     : "Composite",
    bind : { "$base"  : "http://environment.data.gov.uk/water-quality/def/roles" },
    oneOffs : [
        { "@id"             : "<{$base}/>" ,
          "<rdf:type>"      : "<owl:Ontology>",
          "<owl:imports>"   : "<org:>"
        },
        { "@id"             : "<{$base}/scheme>" ,
          "<rdf:type>"      : "<skos:ConceptScheme>",
          "<rdfs:label>"  : "A SKOS ConceptScheme for roles."
        }

    ],
    templates : [
       { "@id"                   : "<{$base}/{role_code.asNumber().format('%02d')}>" ,
         "<rdf:type>"            : "<org:Role>"  ,
         "<rdfs:label>"          : "{role_desc.lang('en')}",
         "<skos:prefLabel>"      : "{role_desc.lang('en')}",
```

```
            "<skos:notation>"       : "{role_code.asNumber().format('%02d')}",
            "<skos:inScheme>"       : "<{$base}/scheme>",
            "<skos:topConceptOf>"   : "<{$base}/scheme>"
        }
    ]
  }
]
```

DCLIB uses an embedded expression language (JEXL) which provides a means to manipulate input values (eg. date and time parsing, creating composite values from multiple fields, syntactic fomatting of literals and the like).

DCLIB is a work in progress and continues to acquire new features.

#### 3.2.1.2 OpenRefine (formerly Google Refine, formerly GridWorks))

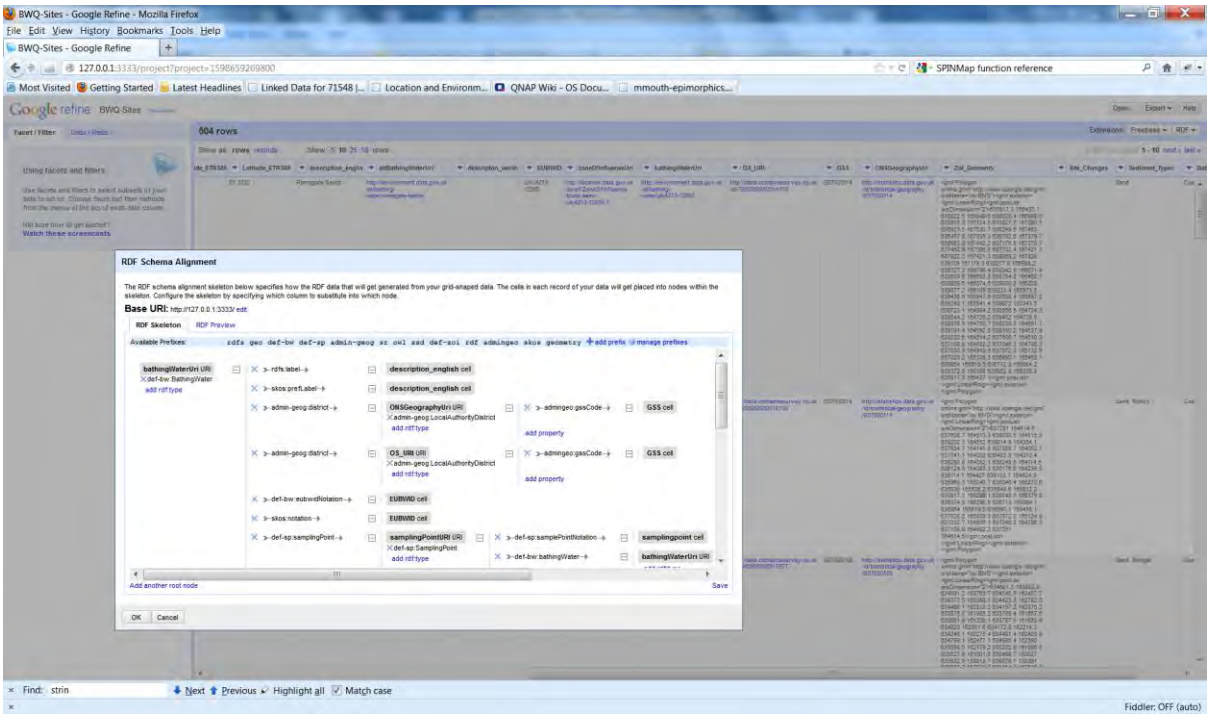
Open Refine can be used to capture a data transformation and apply it to tabular input. it can also be used to clean-up and reorganise tabular data prior to transformation.

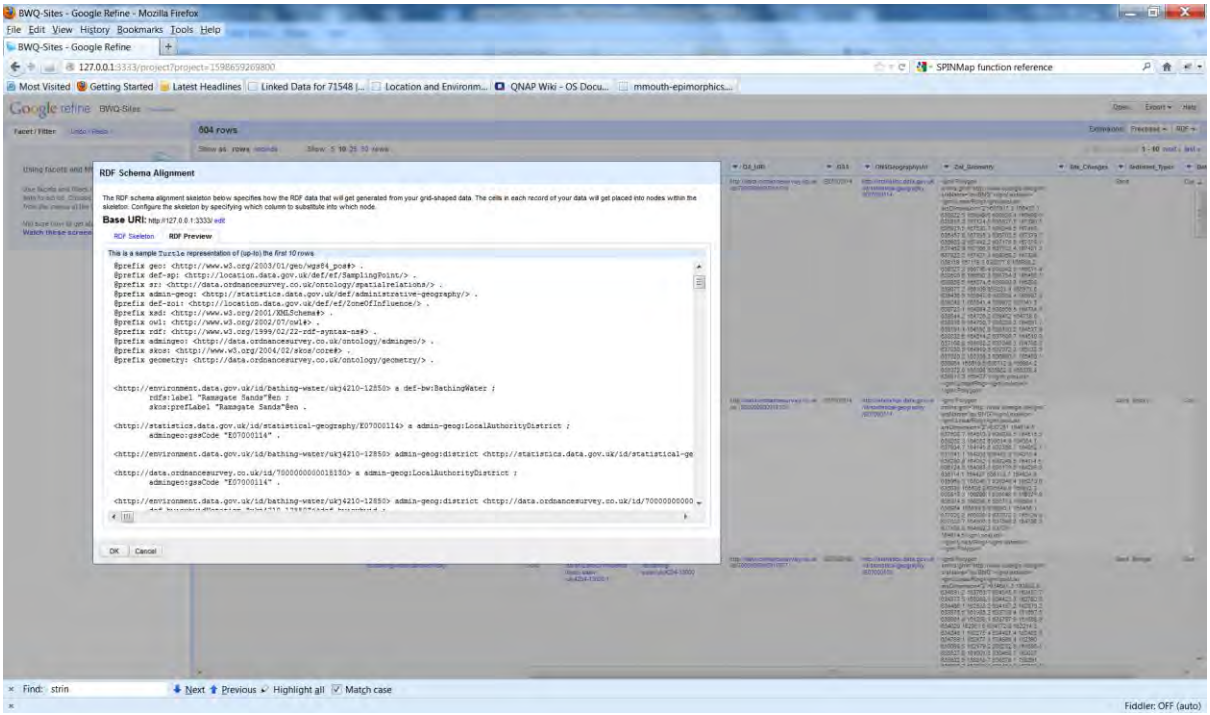
The screenshots below illustrate the use of Open Refine for transforming reference data about bathing-waters and their sampling points.

OpenRefine is a useful tool for developing a data transform in an iterative fashion, however in operation it seems to need to hold the whole data set being transformed in memory - which limits the size of the data tables that can be transformed. It can be a useful way to generate test sets of 'correct' data against which transformation via some other technique can be checked.

The refine framework introduces a key notion of reconciliation, such that references made in one data set can be reconciled against data in a foreign dataset and hence give rise to references made directly with URI from the dataset that the data being transformed is reconciled against.

The screen shots below illustrate an experimental attempt to use OpenRefine to transform reference data for the UK Environment Agency linked data bathing water quality site into RDF.

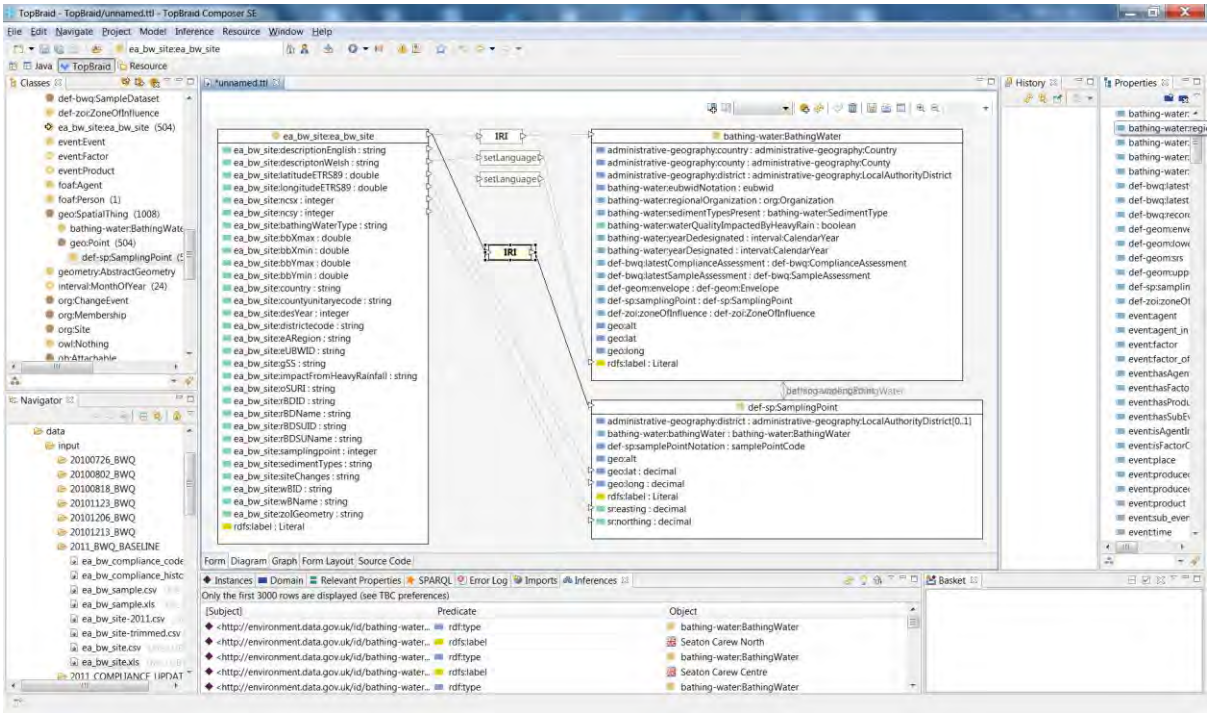





### 3.2.1.3 Topbraid Composer

The Topbraid suite of products from TopQuadrant include facilties to create graphical mapping from a source table to a target vocabulary. There is a function library for manipulating and combining/splitting data from different fields. Not the easiest of tools to drive, but potentially very capable. I believe that the resulting transformations can be exports so that they can used in an automated data conversion workflow. We not tried these facilities on anything other than an experimental basis. The screenshot below is of an attempt

The screen shots below illustrate an experimental attempt to use Topbraid Composer's SPINMap capability to transform reference data for the UK Environment Agency linked data bathing water quality site into RDF.



### 3.2.2 Commercial ETL Tools

We would expect mainstream commercial ETL tools such as FME to have at least plug-ins with RDF generating capability. However, we use predominately open-source tooling and have not made use of commerical ETL tools.

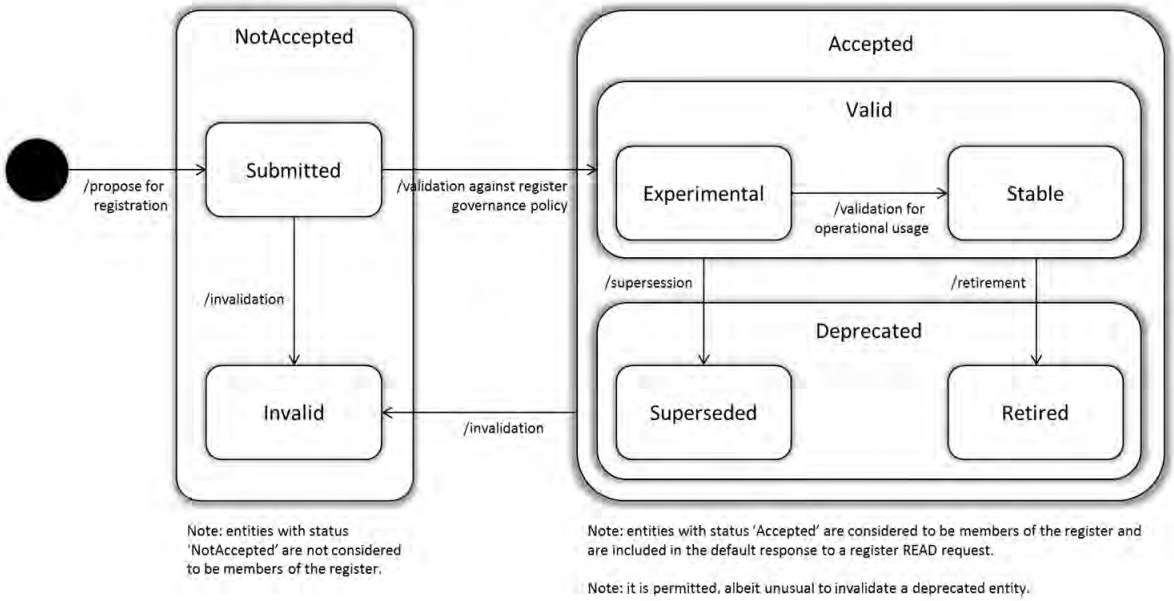
# 4 Architecture and Infrastructure Components

## 4.1 Registers and Registries

At one some level registers are simply lists whose entries are closely managed, and registries are systems that provide the means to implement registers. In part at least they provide the means to mechanise aspects of governance processes. Register items (list entries) can only be made a controlled manner and it is governance processes that exercise that control. The other side of the equation is any operational significance to the presence of an entry in a register.

Registers are typically used express some measure of organisational disposition with respect to the items contained therein - for example the diagram below shows the lifecycle model for a register item adopted by the UKGovLD open source registry (see also https://github.com/UKGovLD/ukl-registry-poc/wiki/Principles-and-concepts)

Register Items typically have a




Registries support the discovery and management of register items within a register. In general registers may be hierarchical (a register of registers... and so-on). In some cases individual register items and the total state of the register at a given instant may be versioned should it be necessary to maintain a history of change in the state of a register. Registered items are typically vocabulary artefacts: classes, properties, codelist and enumerations, application schema and so forth.

The lack of presence of an item in a register can have operational consequences.

- Data may be validated against the contents of one (or more) registers to ensure that the terms used (classes, properties, datatypes, codepoints etc.) are appropriately registered- use of an unregistered term in a document renders the document as invalid.

- Data conversion or mapping tools for mapping from table or data base schema into RDF may consult application-schema specific registers to offer mapping developing guided choices of target class, property or codepoint.

INSPIRE in RDF has a few consequences for registers and registries:

- Registers must be capable of registering the definition of RDF terms (classes, properties, codelist (expressed as skos) and enumerations derived from INSPIRE application schema - and from local member-state extensions.

  It should be able to related those RDF vocabulary terms to the source application-schema from which they were derived. These relations need to be navigable in both directions.

  Alternatively or as well as, the existing INSPIRE registers and feature catalogs need to be extended to include reference to the derived RDF terms - particularly where mapping to pre-existing non-INSPIRE derived vocabulary is used eg. relating say a *ResponsibleParty* to a *foaf:Agent*.

- The registers need to be capable of adopting terms used and defined elsewhere so that their use may be validated. This applies **both**

  - to terms defined in local members state or organisational registers (federation); and

  - to terms whose use is imported from widely used RDF vocabulary (eg. from ORG or FOAF or DC) without themselves ever having been elements of an established application schema (their use is more a consequence of a mapping into RDF) and possibly the application of some local practices.

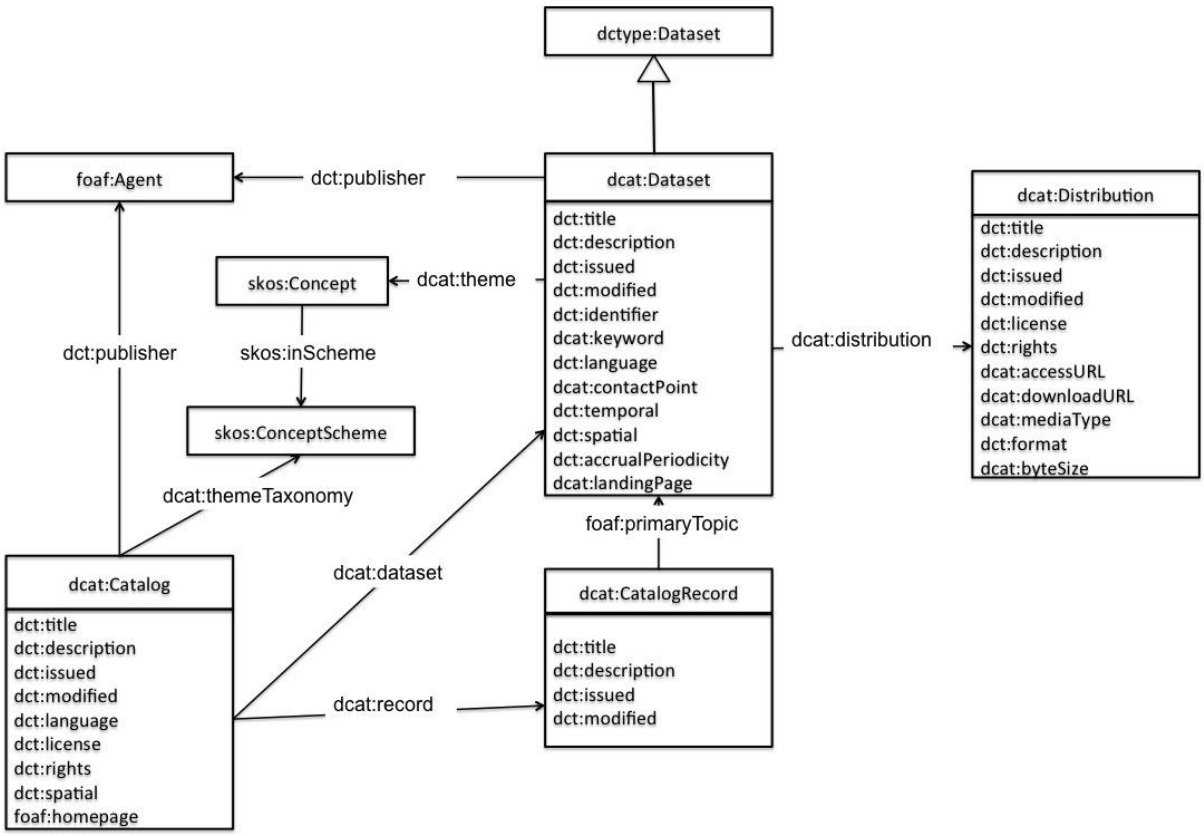
- RDF/linked-data format representations[2] of Register Items and their corresponding Register**ed** items should be available.

## 4.2 Metadata and Catalogues

Further study is needed to understand the relationship and roles of ISO 19115 metadata in contrast to linked data metadata vocabularies such as Dublin Core (DC), Vocabulary of Interlinked Datasets (VoiD) and Data Catalog Vocabulary (DCAT) and whether and how to project ISO 19115 compliant metadata in a form that aligns with VoiD and DCAT, whether to more directly map ISO 19115 constructs into an RDF/OWL vocabulary as has been done for example by Simon Cox (see http://def.seegrid.csiro.au/static/isotc211/iso19115/2003/metadata.ttl)

For RDF/linked data, metadata is just data about a different subject. Where we have a spatial-object (as a graph) that contains data about, say "Manchester Piccadilly Railway Station", and about the spatial-object itself, the information about the railways station is typically referred to as 'the data' and the information about the object (or containing document or graph) is typically referred to as 'the metadata'.

In DCAT, illustrated in the diagram below, notice the separation between datasets (*dcat:Dataset*) and catalog records (*dcat:CatalogRecord*). Note that while they share many Dublin Core properties (*dct:title, dct:description, dct:modified*), in use they each pertain to their respective direct subject ie. dataset or a catalog.



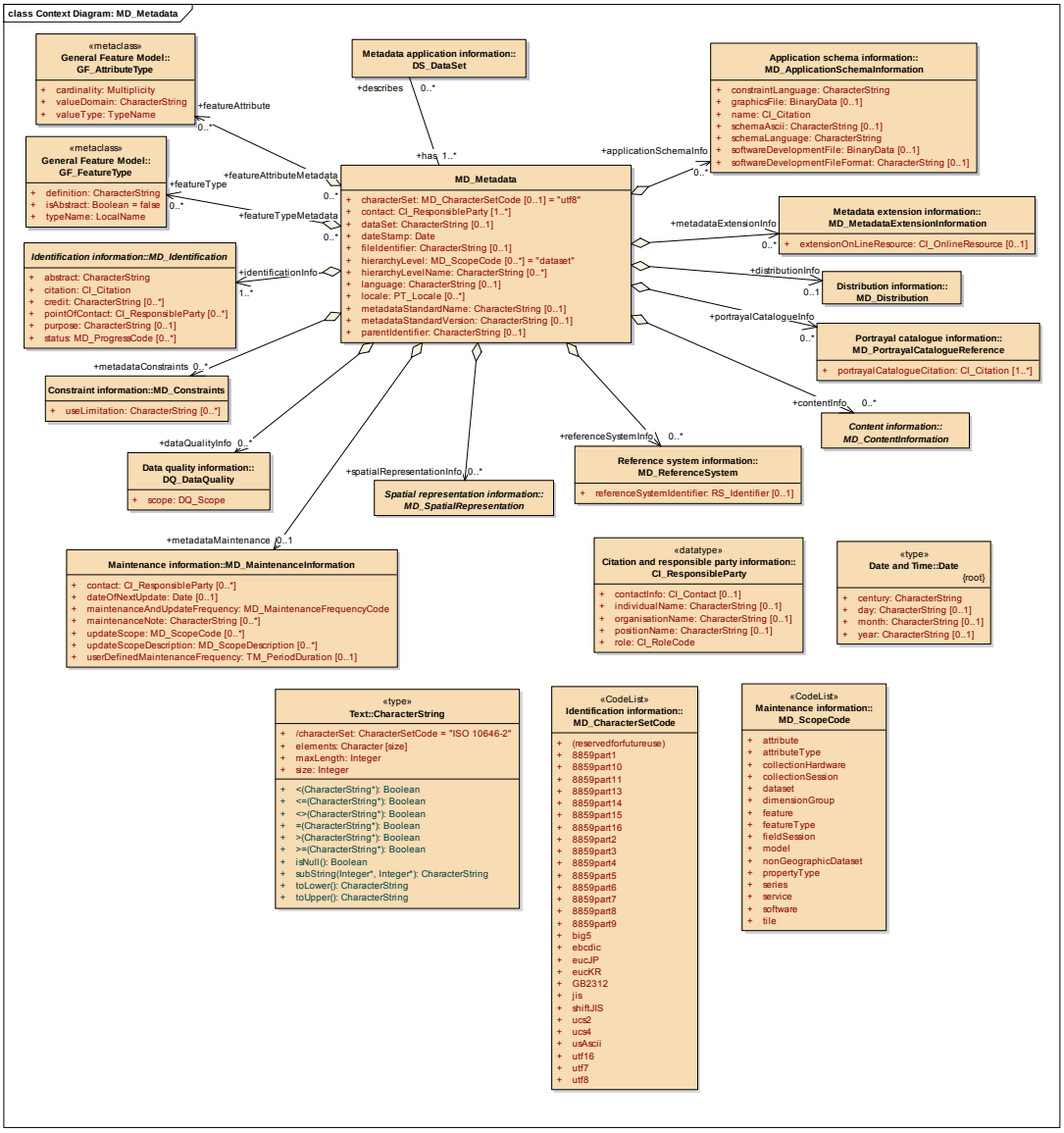

Presenting a catalog record as a named graph in a similar style to our Spatial-Objects as Graphs approach an example catalog record for an example data set might look something like this:

[2] webarch:representation as in http://www.w3.org/TR/webarch/

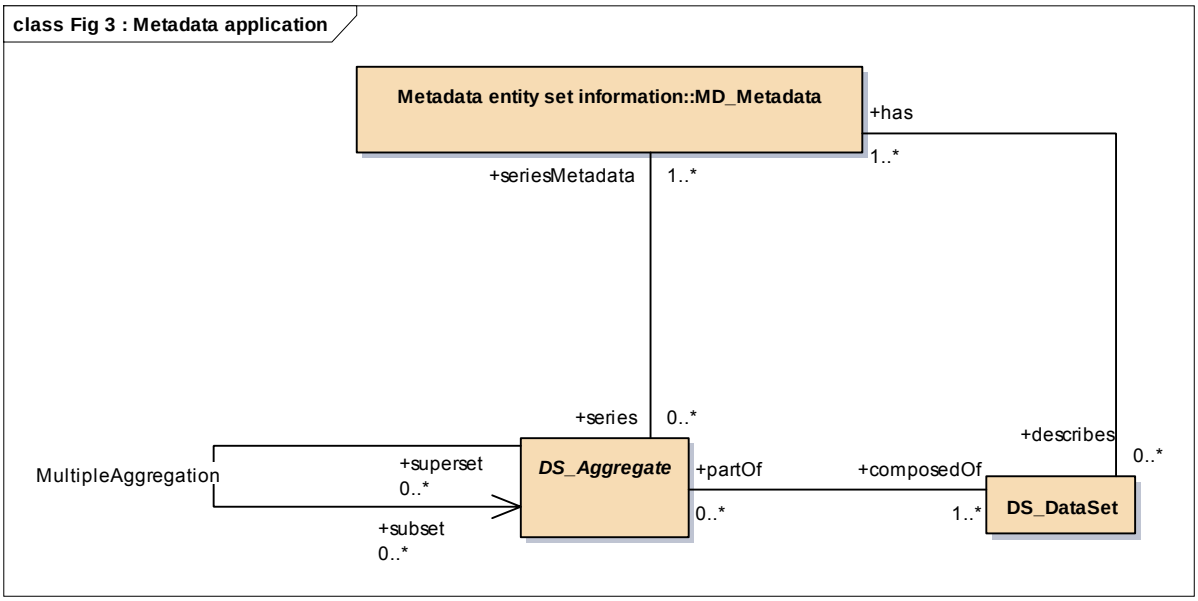
```
ex:dsRecord {
  ex:dataset    a               dcat:Dataset ;
                dct:title       "An example dataset" ;
                dct:modified    "2012-04-01"^^xsd:date
                .

  ex:dsRecord   a               dcat:CatalogRecord ;
                dct:title       "Catalog record for an example dataset"
                foaf:primaryTopic
                                ex:dataset ;
                dct:modified    2014-05-07"^^xsd:date
                .
}
```

Note, for example, that the catalog record was modified more recently than the dataset.

In ISO 19115 the specification sets up a 'describes/has' association between a dataset (*DS_Dataset*) and its metadata records (*MD_Metadata*). The properties of *MD_Metadata* and its satellite aggregates may be making statements about a mixture of subjects and we face the same "what's the subject?" question as was faced when trying to formulate spatial-objects in RDF. The *describes* association role suggest that the properties of *MD_Metadata* 'describe' a *DS_Dataset.* This surfaces the tensions discussed in our earlier deliverable.



The aggregate structure or a data set expressed via *DS_Aggregate* and its *MultipleAggregation* association is typically expressed in linked-data using the VoID vocabulary and setting up *void:subset* relations between a *void:Dataset* and its subordinates




## 4.3 Geospatial Services

Resolving feature URI to spatial-objects is very close in spirit to the ISO/OGC Web Feature Service. For GML representations of spatial-objects a spatial-object URI could be made to resolve into a WFS call for a particular feature. A common pattern of the web is for the truncations of hierarchical URI to list entities for example:

> http://environment.data.gov.uk/id/bathing-water

lists UK bathing waters designated under the EU Bathing Water Directive while:

> http://environment.data.gov.uk/id/bathing-water/ukc2102-03600

is a particular bating water in the North East of England. Similarly parameters added to list URIs can be used to generate filtered responses eg:

> http://environment.data.gov.uk/id/bathing-water?type=LakeBathingWater
>
> http://environment.data.gov.uk/doc/bathing-water?min-samplingPoint.easting=362452&max-samplingPoint.easting=494951&min-samplingPoint.northing=159624&max-samplingPoint.northing=302123

limits responses to bathing waters that are lakes or within a given bounding box.

At least in principle, for GML responses these request URI could be transformed in to WFS calls that return feature collections corresponding to the requested items.

Geospatial support with linked-data stores is becoming more common. Initially it was a differentiator generally for commercial offerings. GeoSPARQL is bringing about some consolidation, although there are also alternative open-source offerings such as Strabon and its stSPARQL variant.

One particular aspect of geospatial data that needs to be addressed some way is the range of spatial reference systems that may be used in a response. In particular, whilst it may be tempting to think of the point coordinates of a points in geometries being store in triplestores (whether as literal valued position lists with multiple coordinates embedded in a literal - or as x-y coordinate property values

expressed as a node representing a point in some RDF graph) it seem unlikely that one would want to materialise geometries within the store in ALL the possible spatial reference systems that might be served by the system. It's more likely that the core of the system will store data using single coordinate reference system.Uploaded data and queries (with spatial filters of some sort) will need coordinate references to be transformed on the way into the system and downloaded data and query responses will need to be transformed on their way out.