

Cookbook

for translating relational data models to RDF schemas

Why moving from relational data models to RDF Schemas

Modelling initiatives usually focus on a Domain Model (a relational data model) that shows the classes, properties and relationships as this greatly aids human understanding of the information space. Domain Models are typically drawn using Unified Modelling Language (UML) class diagrams.

However, to achieve interoperability between two systems that need to seamlessly exchange data, a conceptual Domain Model needs to be implemented in a machine-readable and –understandable format, such as XML or RDF.



RDF published as W3C
recommendation
<http://www.w3.org/RDF>

2004

RDF schemas exist to provide the necessary semantics to enable the correct interpretation of instance data and to facilitate consistency between multiple data publishers.

That said, interoperability is greatly enhanced when data publishers re-use each other's vocabularies and confidence in data is greatly increased when engineers follow best practice.

Our cookbook provides guidance for the person who has the task of translating the Domain Model into an RDF schema.

“The ISA Programme of the European Commission has created a process and methodology for developing shared Domain Models based on a multi-stage process involving a group of experts.”

6 steps for translating a Domain Model into an RDF schema

Start with a robust Domain Model developed following a structured process and methodology.

Research existing terms and their usage and maximise re-use of those terms.

Where new terms can be seen as specialisations of existing terms, create sub class and sub properties as appropriate.

Where new terms are required, create them following commonly agreed best practice in terms of naming conventions etc.

Publish within a highly stable environment designed to be persistent.

Publicise the RDF schema by registering it with relevant services.

Re-use when possible, mint where necessary

Keep in mind that whatever the domain of your vocabulary, **someone else has probably done it already.**

Why re-using existing vocabularies is important?

Re-use greatly aids interoperability.

Use of `dcterms:created`, for example, the value for which should be a data typed date such as `2013-02-21^^xsd:date`, is immediately processable by many machines. If your schema encourages data publishers to use a different term and date format, such as `ex:date "21 February 2013"` – data published using your schema will require further processing to make it the same as everyone else's.

Reuse adds credibility to your schema.

It shows it has been published with care and professionalism, again, this promotes its re-use.

Re-use is easier and cheaper.

Re-using classes and properties from well defined and properly hosted vocabularies avoids your having to replicate that effort.

“It is important to build on, not try to replicate, existing RDF schemas and vocabularies, because this will aid interoperability, will add credibility to your schema, and will decrease development effort and costs.”

Finding existing RDF schemas and vocabularies on Joinup

ISA has launched a new online service to make it easier for public administrations to find and reuse semantic assets. More than 1300 assets from 17 organisations, including several Member States and standardization bodies, can be found via the European Commission Joinup Portal [1].

What is ADMS?

This service is powered by the Asset Description Metadata Schema (ADMS) [2]. This is a standardised metadata vocabulary that helps public administrations, standardisation bodies and other stakeholders to document their semantic assets in a uniformed and structured manner (e.g. name, status, version, where they can be found on the Web, etc).

What are semantic assets?

Semantic assets are highly reusable metadata (e.g. xml schemas, generic data models) and reference data (e.g. code lists, taxonomies, dictionaries, vocabularies) that are used by public administrations, in their information systems, to share information.



The image shows a screenshot of the Joinup search interface. The interface includes a search bar, a list of search results, and various filters. Three numbered annotations are overlaid on the screenshot:

- 1 More focused.** First, Joinup helps you to find interoperability assets available in different websites via a single search.
- 2 More targeted.** Then, Joinup helps you navigate through assets by applying search filters.
- 3 More relevant.** Finally, the search gives you detailed information about the asset including the website from where it can be downloaded.

[1] <http://joinup.ec.europa.eu/>

[2] <https://joinup.ec.europa.eu/asset/adms/description>

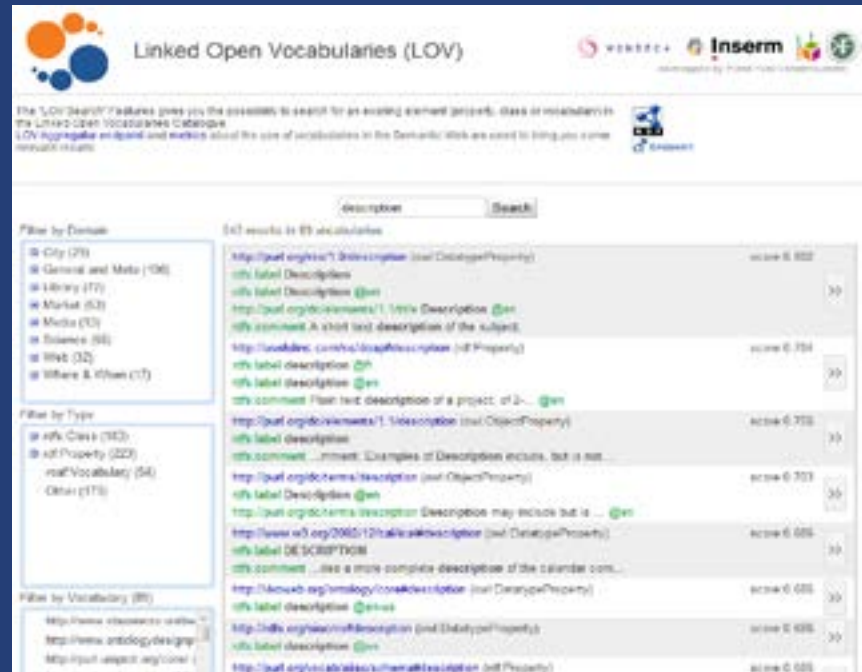
Finding existing RDF schemas and vocabularies on LOV

The Linked Open Vocabularies (LOV) [1] repository, currently associated with the Open Knowledge Foundation, gives a comprehensive view of the available RDF schemas and vocabularies.

It makes them searchable and it is easy to drill down into what you need.

Vocabularies on LOV are described by metadata, classified by vocabulary spaces, and interlinked using Vocabulary of a Friend (VOAF).

LOV allows querying either at vocabulary level or at element level, exploring the vocabulary content using full-text faceted search, and finding metrics about the use of vocabularies in the Semantic Web.



[1] <http://lov.okfn.org>

Creating sub classes and sub properties

RDF schemas and vocabularies often include terms that are very generic. For example, the ORG Ontology's classification property; quoting from the specification [1]:

The ontology does not provide category structures for organization type, organization purpose or roles. Different domains will have different requirements for classification of such concepts. Instead the ontology provides just the core base concepts needed to allow extensions to add specific sub-class structures or classification schemes as required.

*Users of the ontology are encouraged to define **profiles** which strengthen interoperability by specifying particular controlled vocabularies to use for these concepts.*

The Registered Organization vocabulary is an example of such a profile [2]. It defines three sub properties of `org:classification`:

- `companyType`
- `companyStatus`
- `companyActivity`

All three are used to provide different kinds of classification, that is, they are all classifications, but they have tighter semantics than the simple `org:classification` property. Class and sub classes operate in the same way.

“By creating sub class and sub property relationships, systems that understand the super property or super class may be able to interpret the data even if the more specific terms are unknown.”

“Do not create sub classes and sub properties simply to allow you to use your own term for something that already exists. For example, you may want to define a term of 'author' and so be tempted to define a new class of Author as a sub class of `dcterms:Creator`.”

[1] <http://www.w3.org/TR/vocab-org/>

[2] <https://dvcs.w3.org/hg/gld/raw-file/default/legal/index.html>

Minting new terms

If your vocabulary diagram has classes and properties that do not appear in any existing vocabulary in which you have confidence, then of course you need to create the new term.

5 rules for defining new classes and properties

Classes begin with a capital letter and are always singular, e.g. `skos:Concept`.

Properties begin with a lower case letter, e.g. `rdfs:label`.

Object properties should be verbs, e.g. `org:hasSite`.

Data type properties should be nouns, e.g. `dcterms:description`.

Use camel case if a term has more than one word, e.g. `foaf:isPrimaryTopicOf`.

Listing namespaces and adding metadata

First define the namespaces that you are going to use. The ones you will most likely need include:

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
@prefix dcterms: <http://purl.org/dc/terms/>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix adms: <http://www.w3.org/ns/adms#>.
```

It is important that the schema itself includes metadata – that is, data about itself. It is common practice to use the Dublin Core metadata set to provide information about publications. Several terms in the metadata can come from ADMS.

```
<http://purl.org/vocab/cpsv> a owl:Ontology, adms:SemanticAsset;
dcterms:title "Core Public Service Vocabulary"@en;
dcterms:description "The Core Public Service Vocabulary (CPSV) is
designed to make it easy to exchange basic information about the
functions carried out by the public sector and the services in which
those functions are carried out."@en;
dcterms:created "2013-02-06"^^xsd:date;
dcterms:modified "2013-02-24"^^xsd:date;
vann:preferredNamespacePrefix "cpsv";
dcterms:type <http://purl.org/adms/assettype/Ontology>;
dcterms:status <http://purl.org/adms/status/UnderDevelopment>.
```

Tool support

For small schemas, a simple text editor is sufficient, such as Notepad++. Make sure you use UTF-8 encoding, particularly for schemas that involve non-ASCII characters (including accented Latin characters).

If the schema is more complicated then you will need a more specialised tool, such as Top Braid Composer or Protégé. If you use one of these tools then you can be sure that the output will be valid RDF, but if you use a text editor then it's imperative that you validate your work.

You can use the W3C RDF validator for doing that. On top of checking the validity of your schema, it also offers a visualisation – it generates a graph from your schema.

The downside is that it only accepts RDF/XML so, assuming you compose your schema in Turtle, you will need to use the RDF validator of Joshua Tauberer.

```
http://notepad-plus-plus.org/
http://www.topquadrant.com/products/TB_Composer.html
http://protege.stanford.edu/
http://www.w3.org/RDF/Validator/
http://www.rdfabout.com/demo/validator/
```

Defining classes and sub classes

The definition of a class starts by declaring it as an RDF and/or an OWL class. For example,

```
cpsv:PublicService a rdfs:Class, owl:Class;
  rdfs:label "Public Service"@en;
  rdfs:comment "This class represents the service itself. As noted in
    the scope, a public service is the capacity to carry
    out a procedure and exists whether it is used or not.
    It is a set of deeds and acts performed by or on
    behalf of a public agency for the benefit of a citizen,
    a business or another public agency."@en.
```

Importantly, the class has an `rdfs:label` and an `rdfs:comment`. The label is the natural language term itself, e.g. ‘Public Service’ and the comment is the definitive text, e.g. a description or a definition of the class.

Wherever possible, provide the label and definition in multiple languages. For example, the value of `rdfs:label` in Greek would be “Δημόσια Υπηρεσία”@el .

The URI for the term remains `http://purl.org/vocab/cpsv#PublicService` but now it has multilingual labels. It is the URIs that machines care about – labels are just for humans – so although a human may think in terms of Greek, to a machine it's the same as the English language term.

Defining class A as a subclass of a class B requires adding the following triple in its definition.
`classA rdfs:subClassOf classB`

For example, defining that a Registered Organization is a subclass of Formal Organization, should be declared as:

```
rov:RegisteredOrganization a rdfs:Class, owl:Class;
  rdfs:label "Registered Organization"@en;
  rdfs:subClassOf org:FormalOrganization.
```

Defining data type and object properties

Data type properties are properties for which the value is a literal.

For example, the ‘identifier’ datatype property of the ORG ontology is defined as:

```
org:identifier a owl:DatatypeProperty, rdf:Property;  
  rdfs:label "identifier"@en;  
  rdfs:domain org:Organization;  
  rdfs:subPropertyOf skos:notation;
```

As with classes, the property definition may include both an `rdfs:label` and `rdfs:comment`.

Being a property, `org:identifier` begins with a lower case letter. Furthermore, again by convention, as it is a datatype property, it is a noun.

This property has a domain of `org:Organization`. That is, one can infer that the subject of a triple is an instance of the class `org:Organization` where `org:identifier` is the predicate.

Object properties (which can be used for expressing relationships as shown in a Domain Model) are properties for which the value is an RDF or and OWL class.

For example, the ‘Input’ property of the Core Public Service Vocabulary is defined as:

```
cpsv:hasInput a rdf:Property, owl:ObjectProperty;  
  rdfs:label "has input"@en;  
  rdfs:range cpsv:Input;
```

Being an object property – i.e. a relationship between two classes – the label of the ‘Input’ is a verb.

The main difference between object type and data type properties is apparent in the range statement. The range definition here means that one can infer that the object of a triple is an instance of the class `cpsv:Input` where `cpsv:hasInput` is the predicate.

Choosing a namespace and publicising your work

You now need to choose a stable namespace for your RDF schema.

To help you with that, the ISA programme run a study to explore good practices on the publication of persistent Uniform Resource Identifiers (URI) sets, both in terms of format and of their design rules and management.

The conclusions of this study are summarised in the figure below.

“Once your RDF schema is published you will want people to know about it. To reach a wider audience register it on services like Joinup and LOV.”



The infographic is titled "10 rules for persistent URIs". It is divided into two columns: "Good" (marked with a thumbs up) and "Bad" (marked with a thumbs down). The central text "10 rules for persistent URIs" is prominently displayed in the middle.

Good Practices (Thumbs Up)	Bad Practices (Thumbs Down)
Follow the pattern e.g. <code>http://www.isa-ebc.eu/ontology/1.0/ontology.ttl</code>	Avoid stating ownership e.g. <code>http://www.isa-ebc.eu/ontology/1.0/ontology.ttl</code> ❌
Re-use existing identifiers e.g. <code>http://www.isa-ebc.eu/ontology/1.0/ontology.ttl</code>	Avoid version numbers e.g. <code>http://www.isa-ebc.eu/ontology/1.0/ontology.ttl</code> ❌
Link multiple representations e.g. <code>http://www.isa-ebc.eu/ontology/1.0/ontology.ttl</code> e.g. <code>http://www.isa-ebc.eu/ontology/1.0/ontology.ttl</code>	Avoid using auto-increment e.g. <code>http://www.isa-ebc.eu/ontology/1.0/ontology.ttl</code> ❌ e.g. <code>http://www.isa-ebc.eu/ontology/1.0/ontology.ttl</code> ❌
Implement 301 redirects for real-world objects e.g. <code>http://www.isa-ebc.eu/ontology/1.0/ontology.ttl</code>	Avoid query strings e.g. <code>http://www.isa-ebc.eu/ontology/1.0/ontology.ttl</code> ❌
Use a dedicated service e.g. <code>http://www.isa-ebc.eu/ontology/1.0/ontology.ttl</code>	Avoid file extensions e.g. <code>http://www.isa-ebc.eu/ontology/1.0/ontology.ttl</code> ❌

[1] <https://joinup.ec.europa.eu/community/semic/document/10-rules-persistent-uris>

Introduction

Re-use & mint


Creating your RDF schema

Publishing your RDF schema

Good practices

Contact

Good practices for developing RDF schemas



Create sub classes, sub properties and super classes where appropriate.

Offer well defined terms with well designed, persistent URIs.

Publish in multiple formats for consumption by humans and machines.

Ensure that it remains stable for the long term.

Add metadata to make it discoverable.



Do not replicate existing, widely used terms.

Do not add new semantics to existing terms.



[Introduction](#)[Re-use & mint](#)[Creating your RDF schema](#)[Publishing your RDF schema](#)[Good practices](#)[Contact](#)

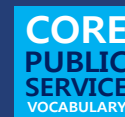
All studies produced by ISA's semantic interoperability initiative are available for download on Joinup through the Semantic Interoperability Community (SEMIC) at

<https://joinup.ec.europa.eu/community/semantic/description>

Contact us

<https://joinup.ec.europa.eu/contact>

Visit our initiatives



Get involved



Follow [@SEMICEu](#) on Twitter



Join [SEMIC](#) group on LinkedIn



Join [SEMIC](#) community on Joinup

Introduction

Re-use & mint

Creating your RDF schema

Publishing your RDF schema

Good practices

Contact

ISA is undertaking a number of initiatives to promote interoperability in Europe



[Introduction](#)[Re-use & mint](#)[Creating your RDF schema](#)[Publishing your RDF schema](#)[Good practices](#)[Contact](#)

Disclaimer

The views expressed in this document are purely those of the writer and may not, in any circumstances, be interpreted as stating an official position of the European Commission. The European Commission does not guarantee the accuracy of the information included in this study, nor does it accept any responsibility for any use thereof. Reference herein to any specific products, specifications, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favouring by the European Commission. All care has been taken by the author to ensure that s/he has obtained, where necessary, permission to use any parts of manuscripts including illustrations, maps, and graphs, on which intellectual property rights already exist from the titular holder(s) of such rights or from her/his or their legal representative.

This Case Study was prepared for the ISA programme by PwC EU Services EESV