



European
Commission

The ISA **programme** What's in it for us?

**Cookbook for Translating
Relational Data Models to RDF
Schemas**

ISA

DOCUMENT METADATA

Property	Value
Release date	27/02/2013
Status:	Acceptance
Version:	0.06
Authors:	Phil Archer W3C Nikolaos Loutas, PwC EU Services Stijn Goedertier, PwC EU Services
Reviewed by:	Joao Frade, PwC EU Services
Approved by:	

DOCUMENT HISTORY

Version	Description	Action
0.01	Creation	Creation
0.02	ToC delivered for review	Review
0.03	Completion of main text	Review
0.04	Updates throughout the document	Update
0.05	Tools section added	Update
0.06	Delivered for acceptance	For acceptance

TABLE OF CONTENTS

SC6DI06692	1
D8.9.1 - Cookbook for Translating Relational Data Models to RDF Schemas	Error! Bookmark not defined.
Deliverable	Error! Bookmark not defined.
Document Metadata	2
Document History	2
Table of Contents	3
List of Figures	4
List of Tables	4
1 Introduction.....	5
1.1 Objectives	5
1.2 Motivation	6
1.3 Scope.....	6
1.4 Overview of the Process.....	7
1.5 Structure	7
2 Re-use When Possible, Mint When Necessary	9
2.1 Existing Key Vocabularies & Their Namespaces	9
2.2 Finding existing vocabularies	14
2.3 Sub classes and sub properties	17
2.4 Minting new terms	18
3 Creating Your Schema – A Worked Example.....	19
3.1 Namespaces and metadata.....	20
3.2 A Simple Class	21
3.3 Multilingualism	22
3.4 Defining a Sub Class	23
3.5 A Data Type Property	23
3.6 An Object Type Property	24
3.7 Domains, Ranges and Inferencing	25
3.8 Describing Your Use of Other People's Terms	26
3.9 Tool support.....	27
4 Publishing Your Schema.....	29
4.1 Choosing a Namespace	29
4.2 Hash or Slash?	30
4.3 Publicise Your Work!	30
5 Summary.....	31
Annex I. Turtle Examples.....	33
Annex II. The Complete CPSV Schema.....	34
References	38

LIST OF FIGURES

Figure 1: Joinup online search service.....	15
Figure 2: Screenshot for 'description' in Linked Open Vocabularies (screenshot taken 2013-02-21)	16
Figure 3: The UML diagram for the Core Public Service Vocabulary	19
Figure 4 Part of the graph of the CPSV produced by the W3C Validator	28
Figure 5: ISA's 10 Rules for persistent URIs	29
Figure 6: Transforming a domain model into an RDF schema	31
Figure 7: Good practices for the development of an RDF schema.....	32

LIST OF TABLES

Table 1 An incomplete list of key vocabularies, some familiarity with which is essential for designing schemas in the public sector.	14
---	----

1 INTRODUCTION

Semantic agreements, such as common vocabularies and schemas, are developed through a multi-stage process, typically by a group of people working to solve a common problem. The ISA Programme of the European Commission has created a process and methodology for developing such semantic agreements [PMDSA], based on good practices set by major standardisation organisations, such as W3C.

The outcome of such modelling initiatives usually focuses 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 such diagrams may not be fully conformant with UML's many powerful and sophisticated features so that that the term 'UML class diagram' should often be interpreted loosely.

In order 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.

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

Schemas exist to provide the necessary semantics to enable the correct interpretation of instance data and to facilitate consistency between multiple data publishers. It is important that schemas are error free. They are a reference point for both machines and human data modellers and it is arguable which of those is the more pedantic.

1.1 OBJECTIVES

This document is designed to help people who wish to create an RDF schema, almost certainly beginning with a Domain Model expressed as a UML class diagram.

The end result should be a schema that:

- does not replicate existing, widely used terms;
- creates sub classes, sub properties and super classes where appropriate;
- does not accidentally add new semantics to existing terms;
- offers well defined terms with well designed, persistent URIs;
- is published in multiple formats for consumption by humans and machines;
- is likely to remain stable for the long term;
- is discoverable.

1.2 MOTIVATION

In a remarkably short time, the world has become used to the World Wide Web. Its success is due largely to its distributed nature and the ability of anyone anywhere to publish their work and to link to other items. Linked data applies the same principles to data – it's how the world can share its data at Web scale.

Important in this vision of shared data is a common approach to modelling data based on vocabularies that, again, anyone can publish. 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. The motivation for this document is therefore to make publishing vocabularies as simple as possible for individuals with some knowledge, but not necessarily specialist knowledge, of the subject.

Linked data is a significant contributor to the ISA Programme's overall aim of *Joining Up Governments*.

This document acts as a companion to ISA's Process and Methodology For Developing Semantic Agreements [PMDSA], expanding significantly on the implementation section relating to the creation and publication of an RDF schema.

1.3 SCOPE

The cookbook assumes a basic knowledge of concepts in vocabulary design such as classes, sub classes, properties and relationships. These concepts are independent of any encoding language used and are fully supported in RDF.

This document is not a general primer on RDF and uses the following terms without further explanation:

- URI
- graph
- triple
- subject
- predicate
- object
- namespace
- prefix

These terms and more are explained in the W3C RDF Primer [PRIMER] and there are many books available on the subject. In Annex I we do, however, provide sufficient guidance on how to read RDF for the purposes of this document.

One aspect that is worth highlighting for the current discussion is that RDF practitioners do not usually talk about relationships and properties (as one would come across in a typical UML class diagram); the equivalent terms are object type property and datatype property respectively (these are defined in the Web Ontology Language [OWL]).

1.4 OVERVIEW OF THE PROCESS

The process of creating an RDF schema for a new Domain Model can be summarised as follows:

1. research existing terms and their usage and maximise re-use of those terms;
2. where new terms can be seen as specialisations of existing terms, create sub class and sub properties as appropriate;
3. where new terms are required, create them following commonly agreed best practice in terms of naming conventions etc.;
4. publish within a highly stable environment designed to be persistent;
5. publicise the vocabulary by registering it with relevant services.

The preceding list is very terse and perhaps over simplified. It assumes that the person tasked with creating the RDF schema is beginning work after development of the vocabulary itself has been completed. This is not an ideal way to proceed – it is much better to include the creation of the RDF schema as part of the vocabulary development process itself. Although a working group will focus its attention on a diagrammatic representation of the vocabulary as it emerges, creating the RDF vocabulary simultaneously avoids any unexpected problems and confusion in the later stages. In particular, re-use of widely known terms can help to ensure that the new terms will be seen as part of the existing landscape and not something entirely new.

1.5 STRUCTURE

The document follows the structure suggested by the bullet points above. I

n section 2 we review existing vocabularies, their key features and how to find more. This leads to a discussion of when it is appropriate to create sub classes and sub properties of existing terms and when one should mint entirely new ones.

In section 3 we work through an example, creating an RDF schema for a real world vocabulary.



Section 4 considers how to actually publish the schema and points to further existing documentation on this topic.

Finally, section 5 concludes the document.

A very brief introduction to writing RDF in Turtle is provided in Annex I.

2 RE-USE WHEN POSSIBLE, MINT WHEN NECESSARY

Vocabulary and ontology design is not a new field - it long pre-dates the World Wide Web - and the chances are that whatever the domain of your vocabulary, someone else has done it already. It's important to build on, not try to replicate, this work. That is, it is important to re-use existing vocabularies. There are several reasons for this.

Firstly, it 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. An individual tasked with processing those dates will almost certainly set out to convert to standard date formats and the Dublin Core `created` term so by using these from the outset, data is immediately more re-usable.

Secondly it adds credibility to your schema – it shows it has been published with care and, again, this promotes its re-use.

Finally, it's easier. The classes and properties in the vocabularies listed in section 2.1, and others, are well defined and properly hosted. Re-using them avoids your having to replicate that effort.

Before creating any new terms in any vocabulary, it is important therefore to make sure that those terms do not already exist. If they do – re-use them!

If something like it already exists but you want to be more specific, create a sub class or sub property. Only if there is nothing that matches the class or property in your vocabulary should you mint a new term.

2.1 EXISTING KEY VOCABULARIES & THEIR NAMESPACES

The ISA programme has created the following vocabularies which are made available as reusable RDF schemas:

- 4 eGovernment core vocabularies¹, namely Core Person, Registered Organisation (originally known as Core Business), Core Location and Core Public Service;
- The Asset Description Metadata Schema (ADMS)²; and

¹ https://joinup.ec.europa.eu/community/core_vocabularies/description

² <https://joinup.ec.europa.eu/asset/adms/release/100>

- The Asset Description Metadata Schema for Open Source Software (ADMS.SW)³.

The table below provides a list of existing vocabularies that anyone creating a new RDF schema ought to be aware of. The danger in providing such a list is that it cannot be complete, there are always other vocabularies that you could re-use and you should seek these out. Conversely, not all of these will be relevant to you. Please see this list as a starting point, not an end.

Name	Usual prefix	Description
RDF	rdf	<p>RDF has some basic properties of its own and terms from this vocabulary appear in almost all RDF data.</p> <p>Base URI http://www.w3.org/1999/02/22-rdf-syntax-ns#</p> <p>Key properties include <code>rdf:type</code> (which can be written as simply 'a' in Turtle).</p> <p>Key class is Resource – the super class for all classes.</p>
RDFS	rdfs	<p>The W3C schema for describing schemas. Terms from this vocabulary are used to define classes, properties, sub classes, sub properties etc.</p> <p>Base URI http://www.w3.org/2000/01/rdf-schema#</p> <p>Key properties include <code>label</code>, <code>comment</code>, <code>description</code>, <code>subClassOf</code>, <code>subPropertyOf</code></p> <p>Key classes are <code>Class</code> and <code>Property</code>.</p>
XML Schema	xsd	<p>Rather than invent its own datatypes, RDF re-uses those defined in XML Schema.</p> <p>Base URI http://www.w3.org/2001/XMLSchema#</p> <p>Key data types include <code>date</code>, <code>dateTime</code>, <code>anyURI</code>, <code>boolean</code>, <code>integer</code>, <code>float</code></p>
Dublin Core	dcterms (sometimes	<p>The Dublin Core Metadata Initiative (named after the city in Ohio where the first meeting was held) is <i>the</i> vocabulary for describing publications and a great deal more. It is highly</p>

³ https://joinup.ec.europa.eu/asset/adms_foss/release/release100

Name	Usual prefix	Description
	dct)	<p>stable and massively implemented.</p> <p>Base URI http://purl.org/dc/terms/</p> <p>Key properties include title, description, creator, created, lastModified.</p> <p>Key classes include Agent, Location,</p>
Web Ontology Language	owl	<p>OWL is a powerful language that encodes formal description logic. Much of its power is out of scope for simple vocabulary design however there are some properties that are very likely to be needed.</p> <p>Base URI http://www.w3.org/2002/07/owl#</p> <p>Key classes include: DatatypeProperty, ObjectProperty</p> <p>Key properties include equivalentClass, equivalentProperty, sameAs.</p>
Friend of a Friend	foaf	<p>Developed simultaneously with the 2004 RDF standards, the FOAF vocabulary is massively used to describe people and their social networks, including by Facebook.</p> <p>Base URI http://xmlns.com/foaf/0.1/</p> <p>Key properties include name, homepage, primaryTopicOf, mbox</p> <p>Key classes include Agent and its sub classes Person, Organization and Group.</p> <p>N.B. foaf:Agent and dcterms:Agent are (OWL) equivalent classes. This means that, for example, all instances of foaf:Person are also instances of dcterms:Agent.</p> <p>Responsibility for maintenance of FOAF rests with two individuals which might raise questions about its long term stability, however, an agreement with the Dublin Core Metadata Initiative is in place such that the latter would take on responsibility in the event of the current owners becoming</p>

Name	Usual prefix	Description
		unwilling or unable to do so [DCMI-FOAF].
Simple Knowledge Organization System	skos	<p>SKOS is used to encode controlled vocabularies, lists etc. Such lists, known as concept schemes in SKOS, can define terms as being equivalent to each other, narrower, broader etc. Outside Concept schemes, SKOS is used extensively to label resources and to provide data-typed literals.</p> <p>Base URI http://www.w3.org/2004/02/skos/core#</p> <p>Key class is skos:Concept</p> <p>Key properties are prefLabel, altLabel and notation.</p> <p>If you need to make comments <i>about</i> the labels, then you need the SKOS-XL extension [SKOSXL]</p>
ORG & RegOrg	org, rov	<p>Originally developed to describe organisations in the UK public sector, the Organization Ontology can be applied to any kind of organisational structure including virtual organisations, commercial bodies and more. It offers mechanisms for describing hierarchies, staff, reporting structures, roles, locations and more.</p> <p>Base URI http://www.w3.org/ns/org#</p> <p>Key classes: Organization, FormalOrganization</p> <p>key properties: hasUnit, classification</p> <p>RegOrg – the Registered Organization Vocabulary, is a profile of ORG that is designed specifically to describe businesses that have legal entity status through a registration process.</p> <p>Key class: RegisteredOrganization</p> <p>key property: registration which links to an adms:Identifier (see below).</p>
The RDF Data Cube	qb	This vocabulary supports the publication of statistics as linked data. They are conceptualised as a hyper cube and

Name	Usual prefix	Description
Vocabulary		<p>the vocabulary is compatible with SDMX [QB].</p> <p>Base URI http://purl.org/linked-data/cube#</p>
Functional Requirements for Bibliographic Records	frbr	<p>This vocabulary is from the library world and is important for differentiating between a conceptual item and its manifestation in the real world. For example, a piece of music can exist as sheet music, a live performance, a particular recording, an individual copy of a recording and so on.</p> <p>Base URI http://purl.org/vocab/frbr/core#</p> <p>Key classes: Work, Expression, Manifestation</p>
The data catalogue vocabulary, DCAT.	dcat	<p>This is the vocabulary used by many open data catalogues around the world, including CKAN and OGPL.</p> <p>Base URI http://www.w3.org/ns/dcat#</p> <p>Key classes: Catalog, Dataset, Distribution</p>
Asset Description Metadata Schema, ADMS	adms	<p>ADMS is similar to DCAT and has many properties in common. However it is designed specifically to describe catalogues of code lists, standards and other 'semantic assets' as opposed to datasets.</p> <p>The Identifier class is important as it allows for descriptions of an identifier – when it was issued and by whom etc.</p> <p>Base URI http://www.w3.org/ns/adms#</p> <p>Key classes: SemanticAssetRepository, SemanticAsset, SemanticAssetDistribution, Identifier</p>
Schema.org	schema	<p>Schema.org is a collaboration between the major search engines and stands apart from other vocabularies in that it duplicates many well known classes and properties within its namespace. It is designed to help search engines make greater sense of otherwise unstructured Web pages. This does not mean that all terms within schema.org are used by the search engines.</p>

Name	Usual prefix	Description
		<p>It takes a deliberately simple approach to information space and so is unlikely to be useful for detailed modelling on its own. However, it provides a handy set of classes and properties that are widely understood and often directly equivalent to those defined elsewhere.</p> <p>Base URI http://schema.org/</p> <p>Key classes: CreativeWork, Event, LocalBusiness etc.</p>

Table 1 An incomplete list of key vocabularies, some familiarity with which is essential for designing schemas in the public sector.

For emphasis, this list is not complete – there will always be more vocabularies that are well used for particular domains but this is a reasonable starting point for public sector information.

NB: We refer the interested reader to the work of Stadtmuller et al. [STAD], who provide a set of metrics indicating the popularity of classes and properties of different vocabularies, based on the Billion Triple Challenge dataset.

2.2 FINDING EXISTING VOCABULARIES

Several services exist for finding existing vocabularies. The European Commission's Joinup platform is one example⁴ and a useful resource specifically for finding existing RDF vocabularies is the Linked Open Vocabularies repository⁵ [LOV].

Joinup, the online service of the European Commission, makes it easier for public administrations to find and re-use 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.

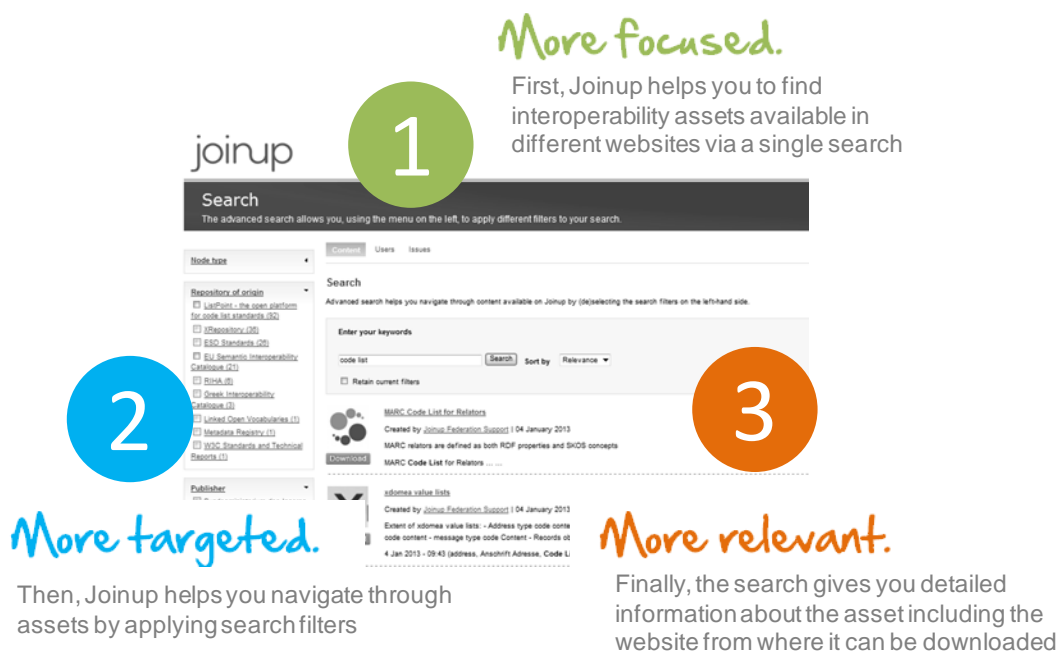
More than one thousand assets from seventeen organisations, including several Member States and standardization bodies, can be found via the European Commission Joinup Portal⁶. By increasing the visibility and promoting the re-use of existing semantic assets the European Commission fosters semantic interoperability among information systems developed in different Member States.

⁴ <http://goo.gl/Ea9bg>

⁵ <http://lov.okfn.org>

⁶ <https://joinup.ec.europa.eu/catalogue/all>

This service is powered by the ADMS, which 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).



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

Figure 1: Joinup online search service

The LOV repository began life in the Data Lift project⁷ and is now associated with the Open Knowledge Foundation. LOV is very useful and gives a comprehensive view of the available vocabularies. It makes them searchable and it's easy to drill down into what you need. Unlike Joinup, which covers a broader spectrum of reusable semantic assets, LOV focuses on Linked Data vocabularies.

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.

The only drawback is that it does not offer any guidance on quality and stability. When deciding whether or not to use a term from an existing vocabulary one must use one's own judgement. There are two key questions to ask:

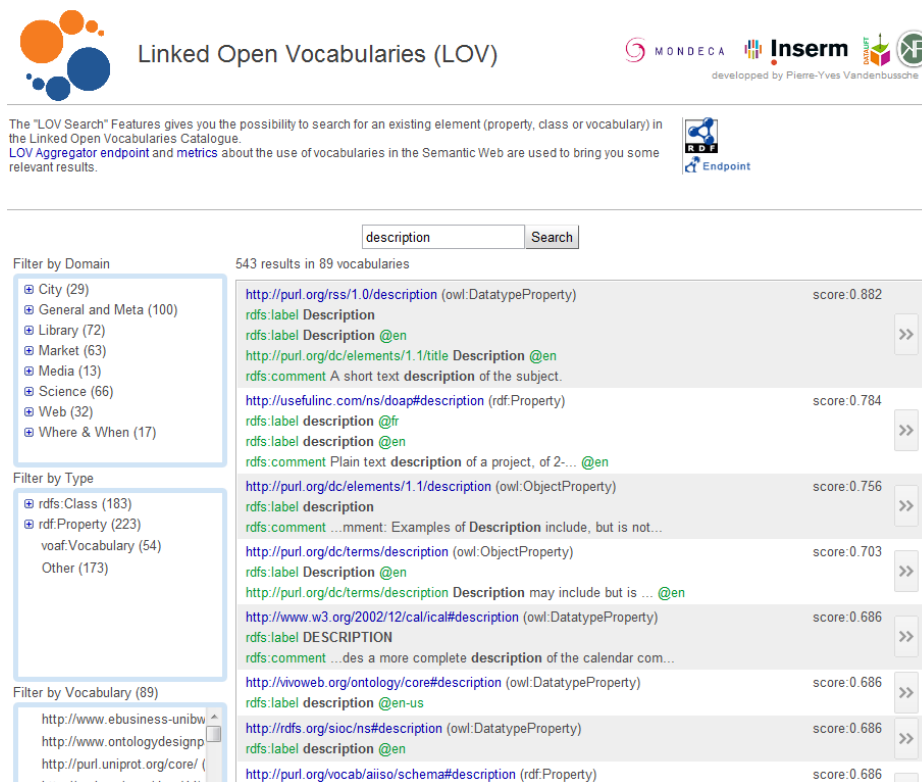
1. is it stable and/or subject to a formal change process?
2. is it already widely used?

⁷ <http://datalift.org/>

All the ones mentioned in the figure below fulfil these criteria as do many others – Good Relations [GR], the Bibliographic Ontology [BIBO] and Creative Commons [CC] for example. It is possible that a vocabulary may be found that appears to fit your needs perfectly but that does not appear to be in widespread use. In this case, the research task is to identify why this is so. Common reasons include:

- the vocabulary is the work of a small number of individuals who did not succeed in building a community around their effort;
- the vocabulary has not been promoted sufficiently;
- the vocabulary is subject to restrictive licence terms and is effectively unavailable for reuse;
- a more popular vocabulary is already in use that does a similar job.

In the first two cases, the authors may welcome an approach from an outside party to help stabilise and promote their work. In the third case, developers of open standards should be wary of infringing copyright and should probably steer well clear. If the fourth case applies then this is a signal that the more popular vocabulary would likely be a better choice. This can be frustrating as it does inevitably lead to compromises being made, however, the aim is to make data interoperable and for that reason *large scale deployment trumps semantic correctness*.



The screenshot shows the LOV search interface. At the top, there are logos for MONDECA, Inserm, and others, with the text 'developped by Pierre-Yves Vandenbussche'. Below the logos, a search bar contains the word 'description' and a 'Search' button. The results are displayed in a table with columns for the URI, the type of property, and a score. The table lists several results, including 'http://purl.org/rss/1.0/description (owl:DatatypeProperty)' with a score of 0.882, 'http://usefulinc.com/ns/doap#description (rdf:Property)' with a score of 0.784, and 'http://www.w3.org/2002/12/cal/ical#description (owl:DatatypeProperty)' with a score of 0.686. On the left side, there are filters for 'Filter by Domain', 'Filter by Type', and 'Filter by Vocabulary'. The 'Filter by Domain' section includes categories like City (29), General and Meta (100), Library (72), Market (63), Media (13), Science (66), Web (32), and Where & When (17). The 'Filter by Type' section includes rdfs:Class (183), rdf:Property (223), voaf:Vocabulary (54), and Other (173). The 'Filter by Vocabulary' section shows a list of URIs, including 'http://www.ebusiness-unibw...' and 'http://www.ontologydesignp...'. The search results are sorted by score, with the highest score at the top.

Figure 2: Screenshot for 'description' in Linked Open Vocabularies (screenshot taken 2013-02-21)

2.3 SUB CLASSES AND SUB PROPERTIES

Vocabularies, such as those listed in Table 1, often include terms that are very generic. For example, the ORG Ontology's `classification` property; quoting from the specification:

*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 [ROV] vocabulary is an example of such a profile. It defines three sub properties of `org:classification`:

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

All three of these 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. All mammals are animals, not all animals are mammals, therefore mammal is a sub class of animal. By creating these 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.

As with simple re-use, defining terms as sub properties and sub classes gives potential users of your schema confidence that you have surveyed the current landscape and have added to it.

It is sometimes tempting to 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`.

Don't..

`dcterms:Creator` is one of the most used properties in linked data and, in data modelling terms, has exactly the same meaning as 'author' in the publishing world. If the semantics match, use the term directly, even if you don't much care for the actual term used. If your semantics are more precise than those in an existing vocabulary, create your sub class/sub property.

2.4 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.

When doing so, bear in mind the following naming conventions:

- properties begin with a lower case letter, e.g. `rdfs:label`;
- use camel case if a term has more than one word, e.g. `foaf:isPrimaryTopicOf`;
- classes begin with a capital letter and are always singular, e.g. `skos:Concept`;
- data type properties should be nouns, e.g. `dcterms:description`;
- object properties should be verbs, e.g. `org:hasSite`.

We'll explore these ideas further in section 3 which works through an example.

3 CREATING YOUR SCHEMA – A WORKED EXAMPLE

Figure 3 shows the UML diagram for the Core Public Service Vocabulary⁸, developed by a working group operating under the ISA Programme. We will work through it to create the RDF schema, noting various decision points along the way. The complete schema is included in Annex II.

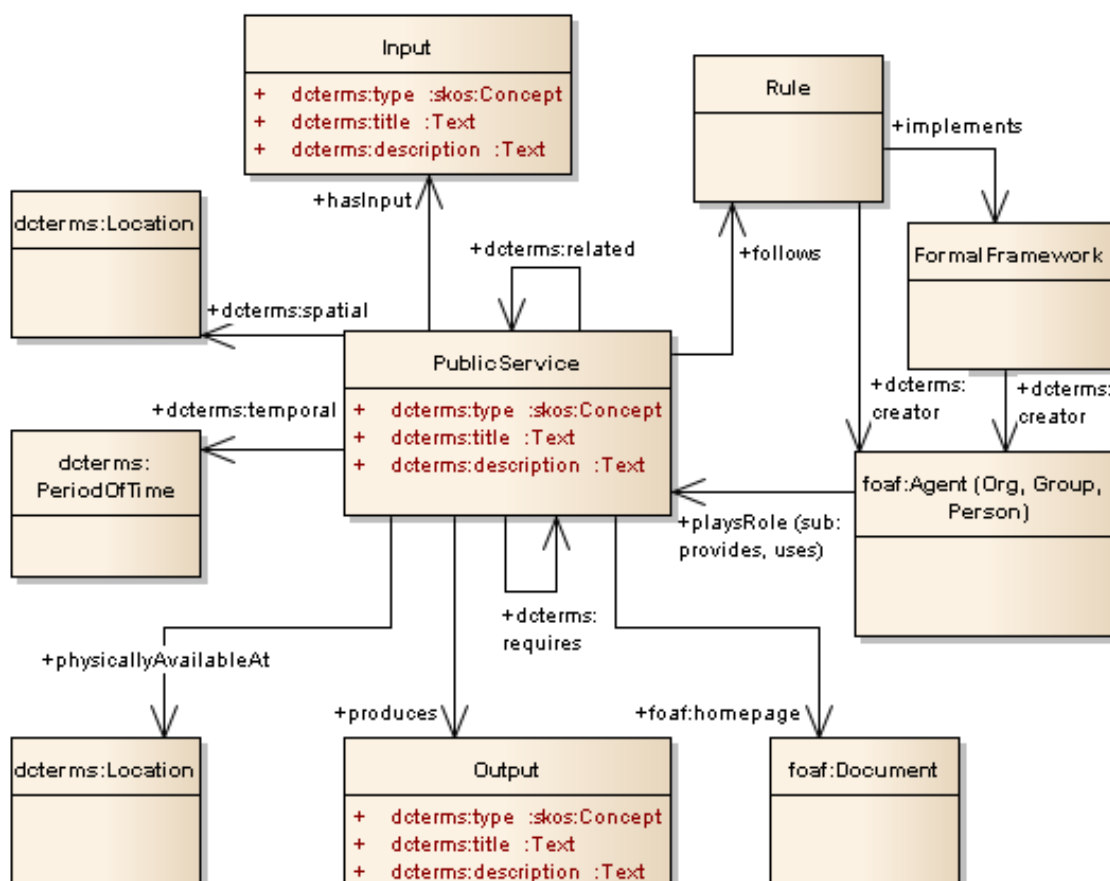


Figure 3: The UML diagram for the Core Public Service Vocabulary

The first thing to notice is that some terms are prefixed, others are not. The text accompanying the diagram states that all terms are in the `cpsv` namespace unless otherwise shown. It is a matter of personal choice whether this pattern is followed or whether you prefer to make all prefixes explicit. Either way, what is clear is that **even within the diagram, the re-use of several terms is already planned.**

⁸ https://joinup.ec.europa.eu/asset/core_public_service/asset_release/core-public-service-vocabulary

The alternative approach is to ignore all other vocabularies until you come to create the schema. This allows you to draw the diagram as you see fit but has the distinct disadvantage that you then end up creating a schema that doesn't appear to match the diagram.

In the CPSV it would have been possible, for example, to include a class of 'Office' where the public service was available. When creating the schema this would have *then* become `dcterms:Location` – something that is very likely to cause confusion. Better to include the existing terms you plan to use in the diagram from the start.

The task ahead of us now is to create the schema for all the classes and properties in the CPSV.

3.1 NAMESPACES AND METADATA

First things first, we will define the namespaces we're going to use:

```
@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 vann: <http://purl.org/vocab/vann/>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix adms: <http://www.w3.org/ns/adms#>.
@prefix frbr: <http://purl.org/vocab/frbr/core#>.
@prefix cpsv: <http://purl.org/vocab/cpsv#>.
```

It's important that the schema itself includes metadata – that is, data about itself.

```
<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";
  foaf:homepage
<http://joinup.ec.europa.eu/asset/core_public_service/description>;
  dcterms:publisher [foaf:name "European Commission"];
  dcterms:creator [foaf:name "Core Public Service Working Group";
  foaf:homepage
```

```
<http://joinup.ec.europa.eu/asset/core_public_service/document/core-  
public-service-working-group>.);  
  dcterms:type <http://purl.org/adms/assettype/Ontology>;  
  dcterms:status <http://purl.org/adms/status/UnderDevelopment>.
```

It is common practice to use the Dublin Core metadata set to provide information about publications. Several terms in the metadata come from ADMS [ADMS] ... which makes extensive use of Dublin Core. The `vann` namespace may be less familiar. It's from *A vocabulary for annotating vocabulary descriptions* [VANN] and has a couple of very useful properties. We use `vann:preferredNamespacePrefix` in the CPSV. Prefixes can be more or less any string of characters but by sticking to the conventional prefixes your schema will be easier to read. There's a really useful prefix lookup service that depends on this common practice⁹. There is no guarantee or requirement that your prefix is unused by others – that's what URIs are for – but of course it's better to choose one that is.

The other useful property in the `vann` namespace is `vann:usageNote` which we'll come back to in section 3.8.

3.2 A SIMPLE CLASS

Time to define our first term. There is no right or wrong order to define terms within the RDF schema – machines don't care – follow whichever patterns works for you, but it is generally best to follow the order in your documentation.

The documentation for the CPSV begins with a definition of the public service class so we will start with that.

```
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;  
  rdfs:isDefinedBy <http://purl.org/vocab/cpsv> .
```

⁹ <http://prefix.cc/>

The definition begins by declaring that `cpsv:PublicService` is both an RDF and an OWL class. The difference between the two is one that need not concern us in this discussion¹⁰. Notice that the term uses camel case and that, as it is a class, it begins with a capital letter. 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.

Where the specification and schema exist as separate documents, the text is usually copied and pasted from one to the other (section 4 has more to say about publishing). The label and comment have both been language tagged. Wherever possible, provide the label and definition in multiple languages.

The final property, `rdfs:isDefinedBy`, points to the schema itself as the defining document. This makes it clear the *this* is the defining schema.

3.3 MULTILINGUALISM

It is an unfortunate truth that most vocabularies are published with labels in a single language. This monolingualism is often part of the reason for the development of entire 'new' vocabularies that nearly match an existing one. If a vocabulary term exists but is not available in your language then you can easily publish new labels and comments. For example, a Greek developer might publish some RDF as follows:

```
cpsv:PublicService
  rdfs:label "Δημόσια Υπηρεσία"@el ;
  rdfs:comment "Η κλάση αυτή αναπαριστά μια δημόσια υπηρεσία. Μια δημόσια υπηρεσία υποδηλώνει την δυνατότητα να εκτελεστεί μια συγκεκριμένη διαδικασία και υπάρχει είτε χρησιμοποιείται είτε όχι. Μια δημόσια υπηρεσία αποτελείται από ένα σύνολο πράξεων που εκτελούνται από ή για λογαριασμό ενός δημόσιου φορέα προς όφελος ενός πολίτη, μιας επιχείρησης (ή οργανισμού) ή ενός άλλου δημοσίου φορέα."@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.

¹⁰ According to the specification of OWL, `owl:Class` is defined as a subclass of `rdfs:Class`. The rationale for having a separate OWL class construct lies in the restrictions on OWL DL (and thus also on OWL Lite), which imply that not all RDFS classes are legal OWL DL classes. In OWL Full these restrictions do not exist and therefore `owl:Class` and `rdfs:Class` are equivalent in OWL Full.

3.4 DEFINING A SUB CLASS

In the CPSV, the Rule and Formal Framework classes are defined as being sub classes of `frbr:Expression` thus:

```
cpsv:Rule a rdfs:Class, owl:Class;
  rdfs:subClassOf frbr:Expression;
  rdfs:label "Rule"@en;
  rdfs:comment "The Rule class represents the specific rules,
  guidelines or procedures that the Public Service follows.
  Instances of the Rule class are FRBR Expressions, that is, a
  concrete expression, such as a document, of the more abstract
  concept of the rules themselves."@en;
  rdfs:isDefinedBy <http://purl.org/vocab/cpsv> .
```

In terms of the RDF this is simple to do using the RDF Schema [RDFS] property `rdfs:subClassOf`. The bigger question perhaps is *why* this was done.

It's because a policy, or a set of guidelines, is a concept. That concept is given form when it is communicated – i.e. expressed. The definition of a `frbr:Expression` is *A class whose members are a realization of a single work usually in a physical form* [FRBRDF]. It means that users of the CPSV can link a Public Service to a document that sets out the rules under which that service operates without implying that the document *defines* those rules. This makes it easier to use the vocabulary whilst remaining true to the semantics – and practicality is important.

3.5 A DATA TYPE PROPERTY

The CPSV does not define any data type properties (properties for which the value is a literal) as Dublin Core provides all it needs, so as an example of a definition of a data type property we will use one from the ORG ontology:

```
org:identifier a owl:DatatypeProperty, rdf:Property;
  rdfs:label "identifier"@en;
  rdfs:label "identifiant"@fr;
  rdfs:domain org:Organization;
  rdfs:subPropertyOf skos:notation;
  rdfs:comment ""Gives an identifier, such as a company
  registration number, that can be used to used to uniquely
  identify the organization. Many different national and
  international identifier schemes are available. The org ontology
  is neutral to which schemes are used. The particular identifier
  scheme should be indicated by the datatype of the identifier
```

```
value. Using datatypes to distinguish the notation scheme used
is consistent with recommended best practice for `skos:notation`
of which this property is a specialization."""@en;
rdfs:comment """Donne un identifiant, comme par exemple le
numéro d'enregistrement d'une entreprise, qui peut être utilisé
comme identifiant unique pour l'Organisation. De nombreux
schémas nationaux et internationaux sont disponibles. Cette
ontologie reste neutre par rapport au schéma utilisé. Le schéma
particulier utilisé devrait être indiqué par le `datatype` de la
valeur de l'identifiant. Utiliser les datatypes pour distinguer
les schémas de notation est cohérent avec les bonnes pratiques
pour `skos:notation` dont cette propriété est une
spécialisation."""@fr;
rdfs:isDefinedBy <http://www.w3.org/ns/org> .
```

Consistent with defining classes as both RDF and OWL classes, the ORG ontology declares properties as RDF properties and either data type or object type properties. As with classes, the property definition includes both an `rdfs:label` and `rdfs:comment`, and the ORG ontology provides a rare example of a schema that is published in multiple languages. Incidentally, the comments are enclosed in triple quotes. This is a feature of Turtle that allows strings to include line breaks.

`org:identifiant` is a property and therefore, by convention, begins with a lower case letter. Furthermore, again by convention, as it is a data type property, it is a noun. It 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:identifiant` is the predicate. See section 3.7 for more on this topic.

3.6 AN OBJECT TYPE PROPERTY

The CPSV defines a number of object properties (relationships). For example, `cpsv:hasInput` is defined thus:

```
cpsv:hasInput a rdf:Property, owl:ObjectProperty;
  rdfs:label "has input"@en;
  rdfs:comment "The hasInput property links a Public Service to
one or more instances of the Input class. A specific service
may require the presence of certain inputs or combinations of
inputs in order to operate. These should be described in an
application profile for a given service."@en;
  rdfs:range cpsv:Input;
  rdfs:isDefinedBy <http://purl.org/vocab/cpsv> .
```


This is very similar to the definition of the data type property seen above – in RDF, the difference between object type and data type properties is only 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. As this is a property, by convention, it begins with a lower case letter. AS it is an object type property – i.e. a relationship between two classes – it is a verb.

3.7 DOMAINS, RANGES AND INFERENCE

As we have seen in the previous two sections, defining domains and ranges for properties allows inferences to be drawn about the nature of the subjects and objects. Many systems make no such inference – it's one of the features of the Semantic Web and, particularly, OWL, that is generally not used in linked data except as described below. *However*, inferences may be drawn by *users*. Even where the machines make no use of inferences, domains and ranges act as a guide to implementers to know how to use your schema to model their data, and to end users to know how to query it.

To take a slightly contrived example: The CPSV makes a distinction between formal frameworks, typically legislation, and rules, i.e. locally set policies and procedures that are derived from that legislation. Hence a public service will follow a set of rules that implement the formal framework. It would be perfectly possible for data that uses the CPSV to describe a public service to use `cpsv:implements` rather than `cpsv:follows` to link a public service to an instance of the Rule class – the world would not stop turning. However, this is not the intention, it breaks the model, and therefore the data is harder to use.

Where inferencing *is* often done within linked data is at ingestion time. That is, when a set of triples is ingested into a triple store, inferred triples may be explicitly generated. This is particularly so for inverse pairs.

Using `owl:inverseOf` it is possible to declare that two properties are inverse pairs. FOAF provides examples of this. Its schema declares

```
foaf:primaryTopic owl:inverseOf foaf:isPrimaryTopicOf.
```

Therefore, for any triple: A `foaf:primaryTopic` B, a triple store may automatically generate a triple stating B `foaf:isPrimaryTopicOf` A.

Likewise, where properties and classes are sub properties and sub classes of others, additional triples may be generated. The properties `cpsv:uses` and `cpsv:provides` are both sub properties of `cpsv:hasRole`, therefore for every triple A `cpsv:uses` B it is also true that A `cpsv:hasRole` B and triple stores may generate this triple at ingestion time. The advantage of

this is that queries that seek "all agents that play a role in the provision and use of a public service" can be quickly answered. That information was present in the original data, the inferred triples just make it easier to access.

Defining domains and ranges within your schema can be very helpful. If over-done, however, it simply restricts use of the vocabulary to a very limited audience and this may be counter productive. You can't make people use your technology in a particular way.

Looking at the `cpsv:hasInput` property again (section 3.6), notice that the range is defined as `cpsv:Input`. That is, in a triple `A cpsv:hasInput B`, we can infer that `B` is 'an input.' However, the domain is not restricted to the `cpsv:PublicService` class as might have been tempting. This is because the concept of something being 'an input' to a process is far wider than just the limited arena of public services. It would therefore be possible something other than a public service to re-use the `cpsv:hasInput`. It is notable that many of the most widely used properties have minimal restriction. The domain of both `dcterms:description` and `skos:prefLabel` is `rdf:Resource` (i.e. anything) and the range is `rdfs:Literal` (any literal, typed or not, language tagged or not). The domain of `foaf:name` is not restricted to `foaf:Person` or even `foaf:Agent` (it's actually defined as `owl:Thing`).

The lesson here is that domains and ranges should be defined where they enhance the semantics of the property but should not be used to impose unnecessary restrictions on its use.

3.8 DESCRIBING YOUR USE OF OTHER PEOPLE'S TERMS

The Dublin Core, FOAF and SKOS terms that appear in the UML class diagram of Figure 3 are all used exactly as those vocabularies define so no further explanation is required within the CPSV schema. However, there may be situations where you want to publish a machine readable statement that says "this is how we use term X in our vocabulary." For example, we may want to make the following statement about the CPSV's use of `dcterms:requires`. In doing so, we do not want to redefine the term, just explain how it is used in this context. Here is how this can be done using `vannUsageNote`:

```
dcterms:requires vann:usageNote "When used in the CPSV to link two Public Services, it means that the operation of one service depends on the successful operation of the other, perhaps by taking the output of one as input to the second. Such dependencies should be defined in the associated Rule. The use of dcterms:requires does not imply any specific details of the dependency."@en.
```

Such a statement achieves our aim of giving information about how the CPSV uses the property but it does not affect the semantics of the Dublin Core term.

You *can* go further and publish domain and range statements about terms in other vocabularies. RDF semantics allow you to do this without affecting the original term, however: no one is obliged to take any notice of your domain and range definitions and, it's worth asking, do you really need to define a domain and range for a term you're re-using?

3.9 TOOL SUPPORT

For small schemas, a simple text editor is sufficient. There are many available such as Notepad++¹¹ and PSPad¹². Make sure you use UTF-8 encoding, particularly for schemas that involve non-ASCII characters (including accented Latin characters). Creating simple schemas such as the CPSV is usually just a matter of copying, pasting and editing elements of an existing schema.

If the schema is more complicated then you'll need a more specialised tool such as Top Braid Composer¹³ or Protégé¹⁴. 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.

There are two key RDF validation tools on the Web. Joshua Tauberer's RDF validator and converter is the essential online service for validating RDF written in Turtle. Simply paste your RDF into the window, select "Notation 3 (or N-Triples/Turtle)" and press Validate! The output of this tool includes the same RDF serialised in RDF/XML as well as the underlying triples.

The other tool is the W3C RDF Validator¹⁵. This has the advantage over the Joshua Tauberer's tool in that it 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'll need to copy and paste the output of Joshua Tauberer's tool into the W3C one.

Visualising the schema is very helpful in showing whether you have your relationships, sub classes and properties all as you imagine them in your head. The graphs it produces are not pretty but they are informative. Figure 4 shows part of the visualisation of the CPSV schema. In the same way that spell checkers make sure that every word you type is a word, a validator just checks your syntax. Whether your schema actually makes sense is another matter and the visualised graph is by far the best way to spot any mistakes. The W3C validator will ensure that

¹¹ <http://notepad-plus-plus.org/>

¹² <http://www.pspad.com/>

¹³ http://www.topquadrant.com/products/TB_Composer.html

¹⁴ <http://protege.stanford.edu/>

¹⁵ <http://www.w3.org/RDF/Validator/>

your use of the `rdf` and `rdfs` namespaces is correct. It's very common to confuse the two and write things like `rdf:comment` (it should be `rdfs:comment`). If you create your schema and find that you have done it perfectly first time then you'll probably be the first person in history to do so.

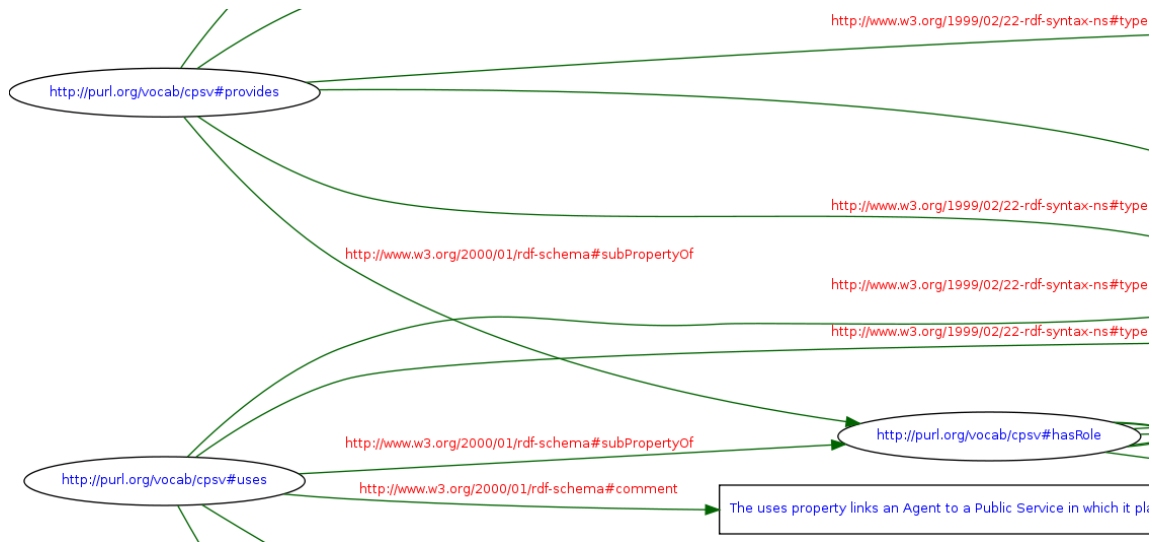


Figure 4 Part of the graph of the CPSV produced by the W3C Validator

4 PUBLISHING YOUR SCHEMA

A schema can be published by simply uploading either a Turtle or RDF/XML file on the Web and, in theory, schemas do not *have* to be dereferencable. However, it *should* be dereferencable at the URI given in the schema and, ideally, it should be available in multiple representations for consumption by both humans and machines.

4.1 CHOOSING A NAMESPACE

The most important factor in choosing a namespace is that it must be stable. For other people to have confidence on your schema they must first be confident that it will still be available in several years' time and that the semantics will remain unchanged (see section 2.2). Do you have access to a stable Web publishing environment? Can you be confident that the namespace you choose will still dereference in 20 years' time? If your organisation can no longer support the namespace, what will happen to it?

These questions are not necessarily easy to answer and there are many factors to bear in mind; hence they are the focus of a separate recent study on URI persistence [URIP] that includes several case studies and guidance notes. The findings of this study are summarised in Figure 5.

In short: URIs and the relevant hosting service must be *designed* for persistence. If no suitable hosting environment is available then services such as purl.org can provide an easy path to stability.



Figure 5: ISA's 10 Rules for persistent URIs

PURL.org¹⁶ is a free online service that can be used for registering and managing persistent URIs. Instead of resolving directly to Web resources, PURLs provide a level of indirection that

¹⁶ <http://purl.org/docs/index.html>

allows the underlying Web addresses of resources to change over time without negatively affecting systems that depend on them. This capability provides continuity of references to network resources that may migrate from machine to machine for business, social or technical reasons.

If you are likely to be publishing multiple vocabularies then it will be worth establishing an environment specifically designed for the purpose. The Neoglism tool produced by DERI and supported by others is specifically designed for this [NEO]. This tool automatically makes your schema available in multiple formats and handles content negotiation to ensure that humans see an HTML document generated from the RDF itself.

4.2 HASH OR SLASH?

In section 3 we looked in detail at the Core Public Service Vocabulary, the namespace for which is `http://purl.org/vocab/cpsv#` - note the final # character. This means that, for example, the full URI for the class `cpsv:PublicService` is `http://purl.org/vocab/cpsv#PublicService`. By contrast, Dublin Core's namespace `http://purl.org/dc/terms/` ends with a / so that `dcterms:creator` is shorthand for `http://purl.org/dc/terms/creator`. It's a matter of personal choice whether to use a # or / character but it does have implications for publishing. Hash is simpler but many individuals prefer the slash method – it's up to you. Both methods are explored in detail in the W3C's Best Practice Recipes for Publishing RDF Vocabularies [BPR].

4.3 PUBLICISE YOUR WORK!

Once your schema is published you will want people to know about it. The community that created the vocabulary will already be aware of it of course but you can reach a wider audience by registering it on services like Joinup and LOV.

5 SUMMARY

The most important aspect of any vocabulary or ontology is that it describes the domain well. That's a job for domain experts. Creating an RDF encoding of that vocabulary involves fitting that domain expertise into the existing landscape of linked data vocabularies. That is a job for someone with experience and knowledge of RDF vocabularies who can recognise that certain concepts, properties and relationships are already well defined and that just need putting together with the 'new' elements. The best results come when the two sets of experts work closely together.

Figure 6 below summarises the steps that need to be taken in order to transform a domain model into an RDF schema, while Figure 7 outlines a set of good practices for the development of RDF schemas.

6 steps for transforming 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.

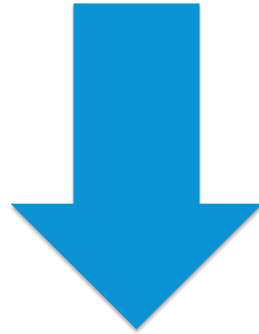
Where new terms are required, create them following commonly agreed best practice in terms of naming conventions etc

Publish within a highly stable environment designed to be persistent.

Publicise the RDF schema by registering it with relevant services.

Figure 6: Transforming a domain model into an RDF schema

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.



Figure 7: Good practices for the development of an RDF schema

ANNEX I. TURTLE EXAMPLES

Examples in the text are given using Turtle as this is most readable RDF syntax, more so than the alternative syntaxes of RDF/XML or n-triples. Details of Turtle are given in its specification [TTL] but the essentials are as follows:

URIs are enclosed in angle brackets thus: `<http://example.com>`.

Compact URIs [CURIE] such as `dcterms:creator` may be used in any position in a triple.

Triples end in a full stop.

Where the same property appears multiple times for the same subject, the values are separated by commas. The most common example of this in schemas is where labels appear in multiple languages. For example:

```
<http://example.com/def/dept>
  skos:prefLabel "Department"@en, "Διεύθυνση"@el.
```

This encodes two triples that declare the preferred English and Greek language label for the same thing.

Different properties and values for the same subject are separated by semi-colons thus:

```
<http://example.com/id/ministry/mareg> a org:FormalOrganization;
  skos:prefLabel "Ministry of Administrative Reform and e-
Governance"@en .
```

This declares two triples: the first states that the thing identified by `http://example.com/id/ministry/mareg` is a Formal Organization, as defined in the `org` namespace; and the second that its preferred English language label is "Ministry of Administrative Reform and e-Governance" (the keyword `a` is a shorthand way of writing `rdf:type`).

ANNEX II. THE COMPLETE CPSV SCHEMA

The Core Public Service Vocabulary defines just 5 classes and 8 properties, all of which are object type properties. The complete schema (correct at the time of writing) is reproduced below. Any subsequent changes will be reflected in the schema itself published at the namespace.

```
@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 vann: <http://purl.org/vocab/vann/>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix adms: <http://www.w3.org/ns/adms#>.
@prefix frbr: <http://purl.org/vocab/frbr/core#>.

@prefix cpsv: <http://purl.org/vocab/cpsv#>.

# metadata

<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";
  foaf:homepage
<http://joinup.ec.europa.eu/asset/core_public_service/description>;
  dcterms:publisher [foaf:name "European Commission"];
  dcterms:creator [foaf:name "Core Public Service Working Group";
foaf:homepage
<http://joinup.ec.europa.eu/asset/core_public_service/document/core-public-
service-working-group>.);
  dcterms:type <http://purl.org/adms/assettype/Ontology>;
  dcterms:status <http://purl.org/adms/status/UnderDevelopment>.

# classes

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;
```

```
    rdfs:isDefinedBy <http://purl.org/vocab/cpsv>.

cpsv:Input a rdfs:Class, owl:Class;
  rdfs:label "Input"@en;
  rdfs:comment "Inputs can be any resource - document, artefact - anything. In a specific context it is likely to be useful to either define a sub class or declare the particular resource to also be of another type as well. A general case might be a foaf:Document but where possible, it is better to refer to a controlled vocabulary of types. dcterms:type should be used to use to provide this information linking to a SKOS Concept."@en;
  rdfs:isDefinedBy <http://purl.org/vocab/cpsv>.

cpsv:Output a rdfs:Class, owl:Class;
  rdfs:label "Output"@en;
  rdfs:comment "Outputs can be any resource - document, artefact - anything. In a specific context it is likely to be useful to either define a sub class or declare the particular resource to also be of another type as well. A general case might be a foaf:Document but where possible, it is better to refer to a controlled vocabulary of types. dcterms:type should be used to use to provide this information linking to a SKOS Concept."@en;
  rdfs:isDefinedBy <http://purl.org/vocab/cpsv>.

cpsv:Rule a rdfs:Class, owl:Class;
  rdfs:subClassOf frbr:Expression;
  rdfs:label "Rule"@en;
  rdfs:comment "The Rule class represents the specific rules, guidelines or procedures that the Public Service follows. Instances of the Rule class are FRBR Expressions, that is, a concrete expression, such as a document, of the more abstract concept of the rules themselves."@en;
  rdfs:isDefinedBy <http://purl.org/vocab/cpsv>.

cpsv:FormalFramework a rdfs:Class, owl:Class;
  rdfs:subClassOf frbr:Expression;
  rdfs:label "This class represents the legislation, policy or policies that lie behind the rules that govern the service. As with the Rule class, the Formal Framework class is a sub class of frbr:Expression, i.e. instances of the class are concrete expressions of the more abstract concept of the piece of legislation or policy itself."@en;
  rdfs:isDefinedBy <http://purl.org/vocab/cpsv>.

# properties (all of which are object type properties)

cpsv:physicallyAvailableAt a rdf:Property, owl:ObjectProperty;
  rdfs:label "physically available at"@en;
  rdfs:comment "A physical location at which a user may interact with the Public Service."@en;
  rdfs:domain cpsv:PublicService;
  rdfs:range dcterms:Location;
  rdfs:isDefinedBy <http://purl.org/vocab/cpsv>.

cpsv:hasInput a rdf:Property, owl:ObjectProperty;
```

```
rdfs:label "has input"@en;
rdfs:comment "The hasInput property links a Public Service to one or more
instances of the Input class. A specific service may require the presence of
certain inputs or combinations of inputs in order to operate. These should
be described in an application profile for a given service."@en;
rdfs:range cpsv:Input;
rdfs:isDefinedBy <http://purl.org/vocab/cpsv>.
# No domain defined as this would hinder re-use of the property
unnecessarily.

cpsv:produces a rdf:Property, owl:ObjectProperty;
  rdfs:label "produces"@en;
  rdfs:comment "The produces property links a Public Service to one or more
instances of the Output class which is its range."@en;
  rdfs:range cpsv:Output;
  rdfs:isDefinedBy <http://purl.org/vocab/cpsv>.
# No domain defined as this would hinder re-use of the property
unnecessarily.

cpsv:implements a rdf:Property, owl:ObjectProperty;
  rdfs:label "implements"@en;
  rdfs:comment "The implements property links a Rule to relevant legislation
or policy documents i.e. the formal framework under which the Rules are
defined."@en;
  rdfs:domain cpsv:Rule;
  rdfs:range cpsv:FormalFramework;
  rdfs:isDefinedBy <http://purl.org/vocab/cpsv>.

cpsv:hasRole a rdf:Property, owl:ObjectProperty;
  rdfs:label "has role"@en;
  rdfs:comment "This very general property links an Agent to a Public Service
in which it plays some role. Both 'provides' and 'uses' are sub properties
of playsRole with specific semantics."@en;
  rdfs:domain dcterms:Agent;
  rdfs:range cpsv:PublicService;
  rdfs:isDefinedBy <http://purl.org/vocab/cpsv>.

cpsv:provides a rdf:Property, owl:ObjectProperty;
  rdfs:label "provides"@en;
  rdfs:comment "The provides property links an Agent to a Public Service for
which it is responsible. Whether it provides the service directly or
outsources it is not relevant, the Agent that provides the service is the
one that is ultimately responsible for its provision."@en;
  rdfs:subPropertyOf cpsv:hasRole;
  rdfs:isDefinedBy <http://purl.org/vocab/cpsv>.

cpsv:uses a rdf:Property, owl:ObjectProperty;
  rdfs:label "uses"@en;
  rdfs:comment "The uses property links an Agent to a Public Service in which
it plays the specific role of user, meaning that it provides the input and
receives the output but does not play any direct role in providing the
```



service. This will typically be an individual citizen or an outside organisation."@en;

rdfs:subPropertyOf cpsv:hasRole;

rdfs:isDefinedBy <<http://purl.org/vocab/cpsv>>.

cpsv:follows a rdf:Property, owl:ObjectProperty;

rdfs:label "follows"@en;

rdfs:comment "The follows property links a Public Service to the Rule(s) under which it operates."@en;

rdfs:domain cpsv:PublicService;

rdfs:range cpsv:Rule;

rdfs:isDefinedBy <<http://purl.org/vocab/cpsv>>.

REFERENCES

- [A11] Action 1.1 Improving semantic interoperability in European eGovernment systems http://ec.europa.eu/isa/actions/01-trusted-information-exchange/1-1action_en.htm
- [ADMS] Asset Description Metadata Schema, Makx Dekkers (Editor), PwC EU Services, <http://joinup.ec.europa.eu/asset/adms/home>
- [BIBO] The Bibliographic Ontology, Frédérick Giasson and Bruce D'Arcus. <http://bibliontology.com/>
- [BR] Best Practice Recipes for Publishing RDF Vocabularies, D. Berrueta, J. Phipps, W3C Note 2008. <http://www.w3.org/TR/swbp-vocab-pub/>
- [CC] Creative Commons Rights Expression Language <http://creativecommons.org/ns#>
- [CPSV] Core Public Service Vocabulary, European Commission/ISA Programme http://joinup.ec.europa.eu/asset/core_public_service/asset_release/core-public-service-vocabulary
- [CURIE] CURIE Syntax 1.0, A syntax for expressing Compact URIs, M. Birbeck, S. McCarron, W3C Working Group Note, 2010. <http://www.w3.org/TR/curie/>
- [DC] DCMI Metadata Terms, Dublin Core Metadata Initiative. <http://dublincore.org/documents/dcmi-terms/>
- [DCAT] Data Catalog Vocabulary (DCAT), F. Maali, P. Archer, J. Erickson. W3C Recommendations track <http://www.w3.org/TR/vocab-dcat/>
- [DCMI-FOAF] Agreement between DCMI and the FOAF Project, 2011 <http://dublincore.org/documents/dcmi-foaf/>
- [FOAF] Friend of a Friend <http://xmlns.com/foaf/spec/>
- [FRBR] Functional Requirements for Bibliographic Records, IFLA 1998. <http://www.ifla.org/publications/functional-requirements-for-bibliographic-records>
- [FRBRDF] Expression of Core FRBR Concepts in RDF. I. Davis, R. Newman, 2005. <http://vocab.org/frbr/core>
- [GR] Good Relations Vocabulary, Martin Hepp <http://purl.org/goodrelations/>

-
- [ISA] Interoperability Solutions for European Public Administrations, <http://ec.europa.eu/isa/>
- [LOV] Linked Open Vocabularies, Barnard Vatant and Pierre-Yves Vandenbussche <http://lov.okfn.org/>
- [NEO] Neologism Web-based RDF Schema vocabulary editor and publishing system. See <http://neologism.deri.ie/>
- [ORG] An Organization Ontology, Dave Reynolds, October 2010. Soon to be republished by W3C at <http://www.w3.org/TR/gld-org/>
- [OWL] OWL 2 Web Ontology Language Document Overview (Second Edition). W3C OWL Working Group. W3C Recommendation 2012 <http://www.w3.org/TR/owl2-overview/>
- [PMDSA] Process And Methodology For Developing Semantic Agreements, M.Dekkers, N. Loutas, S. Goedertier, N. Van Hee, 2013. Under development for publication on Joinup.
- [PMDCV] Process and Methodology for Developing Core Vocabularies, 22 November 2011. <https://joinup.ec.europa.eu/elibrary/document/isa-deliverable-process-and-methodology-developing-core-vocabularies>
- [QB] The RDF Data Cube Vocabulary, D. Reynolds, R. Cyganiak, W3C Recommendations track <http://www.w3.org/TR/vocab-data-cube/>
- [RDF] RDF Primer, F. Manola, E. Miller W3C Recommendation 2004 <http://www.w3.org/TR/rdf-primer/>
- [RDFS] RDF Vocabulary Description Language 1.0: RDF Schema D. Brickley, RV Guha. W3C Recommendation 2004 <http://www.w3.org/TR/rdf-schema/>
- [RFC 3066] H. Alvestrand, ed. RFC 3066: Tags for the Identification of Languages 1995. Available at: <http://www.ietf.org/rfc/rfc3066.txt>
- [ROV] The Registered Organization Vocabulary, P. Archer, A. Papantoniou, W3C Recommendations Track, 2013 <http://www.w3.org/TR/vocab-regorg/>
- [SKOS] SKOS Simple Knowledge Organization System Reference. A. Miles, S. Bechofer. W3C Recommendation 2009. <http://www.w3.org/TR/skos-reference/>
- [SKOSXL] SKOS Simple Knowledge Organization System eXtension for Labels (SKOS-XL), W3C 2009. <http://www.w3.org/TR/skos-reference/skos-xl.html>

- [STAD] Accessing information about Linked Data vocabularies with vocab.cc, S. Stadtmuller, A. Harth, M. Grobelnik, Joint Conference of 6th Chinese Semantic Web Symposium and the First Chinese Web Science Conference, 2012
<http://www.planet-data.eu/sites/default/files/publications/Accessing-Information-about-Linked-Data-Vocabularies-with-vocab.pdf>
- [TOGM] Towards Open Government Metadata, V. Peristeras, DG DIGIT, ISA Unit, September 2011
https://joinup.ec.europa.eu/sites/default/files/towards_open_government_metadata_0.pdf
- [TTL] Turtle Terse RDF Triple Language. E. Prud'hommeaux, G. Carothers (Eds), W3C Recommendation track document, 2013. <http://www.w3.org/TR/turtle/>
- [URIP] Study on persistent URIs, with identification of best practices and recommendations on the topic for the MSs and the EC. P. Archer, S. Goedertier, N. Loutas, 2012. See <http://joinup.ec.europa.eu/community/semic/document/10-rules-persistent-uris>
- [VANN] A vocabulary for annotating vocabulary descriptions, I. Davis, 2005.
<http://vocab.org/vann/>
- [XSD] XML Schema Part 2: Datatypes Second Edition. W3C Recommendation 28 October 2004. <http://www.w3.org/TR/xmlschema-2/#date>