# Best Practices for Implementing FAIR Vocabularies and Ontologies on the Web

Daniel Garijo[1][0000−0003−0454−7145] and
María Poveda-Villalón[2][0000−0003−3587−0367]

[1] Information Sciences Institute, University of Southern California
dgarijo@isi.edu
[2] Ontology Engineering Group, Universidad Politécnica de Madrid
mpoveda@fi.upm.es

**Abstract.** With the adoption of Semantic Web technologies, an increasing number of vocabularies and ontologies have been developed in different domains, ranging from Biology to Agronomy or Geosciences. However, many of these ontologies are still difficult to find, access and understand by researchers due to a lack of documentation, URI resolving issues, versioning problems, etc. In this chapter we describe guidelines and best practices for creating accessible, understandable and reusable ontologies on the Web, using standard practices and pointing to existing tools and frameworks developed by the Semantic Web community. We illustrate our guidelines with concrete examples, in order to help researchers implement these practices in their future vocabularies.

**Keywords:** Ontology metadata · Ontology publication · Ontology access · FAIR principles · Linked Data principles.

## 1 Introduction

In the last decade, a series of initiatives for open data, transparency and open science have led to the development of a myriad of datasets and linked Knowledge Graphs on the Web.[3] Ontologies and vocabularies have been developed accordingly to represent the contents of these datasets and Knowledge Graphs and help in their integration and linking. However, while significant effort has been spent on making data Findable, Accessible, Interoperable and Reusable (FAIR) [18], ontologies and vocabularies are often difficult to access, understand and reuse. This may be due to several reasons, including a lack of definitions of ontology classes and properties; deprecated or unavailable imported ontologies, non-resolvable ontology URIs, lack of examples and diagrams in the documentation, or having scientific publications describing an ontology without any reference to its implementation.

The scientific community has started to acknowledge the need for ontologies to be properly documented, versioned, published and maintained following the

[3] https://lod-cloud.net/

Linked Data Principles [6] and adapting the FAIR principles for data [8]. But these recommendations do not include guidelines on how to implement them for a target vocabulary. In this chapter we address this issue by describing how to make an ontology or vocabulary comply with the FAIR principles, including examples summarizing best practices from the community and our own experience; and pointing to popular existing tools and frameworks.

Our guidelines are aimed at ontology engineers, and therefore the chapter is structured following an ontology development process: Section 2 describes design decisions to be considered when creating an ontology URI (naming conventions, versioning, permanent URIs); Section 3 describes how to create a documentation that is easy to reuse and understand by others (with minimum metadata and diagrams); Section 4 illustrates how to make an ontology accessible and findable on the Web; Section 5 points to existing end-to-end frameworks that support the ontology publication process; and Section 6 concludes our chapter. We consider the design and development of an ontology out of the scope of this chapter, as it has been covered by existing methodologies (e.g., LOT[4] or NeOn [14]).

## 2 Accessible Ontology URI Design

Ontologies are digital artifacts, and therefore they should follow the Linked Data Principles,[5] and use a URI namespace under control of its authors. The rationale is simple: only in a domain under our control we will be able to serve the right serialization of our ontology.

When creating an ontology, it is also important to think carefully about its name, namespace prefix and URI design. Well engineered ontologies are costly to produce, and therefore they should be accessible to other potential users (e.g., by avoiding complex URIs) and easy to differentiate from existing vocabularies.

In this section we describe our proposed best practices for designing URIs for ontologies to make them unique, easy to remember, and easy to maintain. We acknowledge that naming conventions may be subjective, but these practices reflect our experience in ontology development over a wide range of domains (smart cities, open science, meteorology, neuroscience, etc.) and therefore provide a good reference for others. We divide our guidelines in five main key points for accessible ontology URI design: how to select a name and prefix (Section 2.1), whether to use hash or slash namespaces (Section 2.2), whether to use opaque URIs (Section 2.3), how to incorporate semantic versioning in an ontology (Section 2.4) and how to make an ontology URI permanent (Section 2.5).

For illustrating purposes, we will be using an example ontology throughout this chapter, with the following URI: `https://w3id.org/example#`.

### 2.1 Ontology name and namespace prefix

The name and prefix of an ontology are in most cases related to its application domain. **Short** and **simple** names will help others remember your ontology

[4] `https://lot.linkeddata.es/`
[5] `https://www.w3.org/DesignIssues/LinkedData.html`

easily. An extended practice for prefixes is to **use acronyms** to abbreviate the name of longer ontologies, as in "The Data Catalog Ontology"[6] (with prefix **dcat**) or the "Friend of a Friend Ontology"[7] (with prefix **foaf**).

Another aspect to consider when designing the name of your ontology is to avoid overlapping with other existing vocabularies. While it is possible to overload existing prefixes by assigning them a different URI, this often confuses potential re-users that are already familiar with the state of the art. A good strategy to prevent this problem is to look for existing prefixes in common vocabulary registries such as prefix.cc,[8] Linked Open Vocabularies (LOV) [15] or Bioportal [17]. In our example ontology, by doing a quick search in LOV and Bioportal, we can see that our example ontology URI (`https://w3id.org/example`) is not being used. However, the prefix "example" has already been registered to refer to "example.org", a namespace defined to create sample URIs. Therefore we decide on **exo** (derived from *example ontology*) as our namespace prefix.

### 2.2 Hash versus slash URIs

When designing the URI of an ontology, it is important to determine whether the trailing element will be a hash ("#") or a slash ("/"). On the one hand, using a hash makes serving the ontology easier,[9] as the client looking at the URI strips only the part of the URI before the hash symbol. Everything after the hash symbol is interpreted as a *fragment identifier*, and may be ignored or used by browsers to refer to the right section of the HTML documentation (if available). W3C standards usually follow the hash convention, and we will follow it in our example as well.

On the other hand, using a slash allows treating each element of the ontology as a separate entity that may be described in an independent manner. This can be useful for organizational purposes when the ontology is large (e.g., thousands of classes) and returning a full serialization or rendering a single HTML documentation may be deemed too slow, showing instead each class and property separately. The Open Biomedical Ontology network[10] is an example that follows this convention.

### 2.3 Opaque URIs for classes and properties

Opaque URIs obfuscate the name of classes and properties that are part of your ontology. For instance, let us assume we have a "Researcher" class in our example ontology. In order to use opaque URIs, instead of using "Researcher" we would associate the class with an identifier such as "EXO_C0001", and use it as part of its URI. This has two main advantages: First, if we decide to change the name of

[6] `https://www.w3.org/TR/vocab-dcat-2/`
[7] `http://xmlns.com/foaf/spec/`
[8] `http://prefix.cc`
[9] `https://www.w3.org/wiki/HashVsSlash`
[10] `http://www.obofoundry.org/`

the class in the future, the URI of the class would not be affected by it. Second, identifiers are language agnostic. For instance, someone using another alphabet (e.g., chinese, cyrillic, etc.) would be able to refer to the same URI with the corresponding label. Examples of ontologies that follow this convention can be found in the Open Biomedical Ontologies, but it is also followed by commonly used Knowledge Graphs such as Wikidata [16].

The main drawback of using opaque URIs is the difficulty of interpreting properly classes and properties, which usually requires additional tooling for displaying the right labels. For this reason, we will not use them in our example.

### 2.4 Ontology versioning

Ontologies often have multiple versions, and these should be appropriately annotated as part of the ontology metadata (e.g., with the property *owl:versionIRI* and *owl:versionInfo*). We recommend using *semantic versioning*[11] as a guideline for identifying the different versions of an ontology, as it has become a common practice in software engineering. In semantic versioning, each version identifier should follow the format **X.Y.Z**, where **X** represents a major version (e.g., defining a set of classes and properties to support new use cases), **Y** represents a minor version (e.g., adding a single property or class), and **Z** represents patches or quick bug fixes (updating a label, adding examples, etc.). In our example ontology, the first version is **1.0.0** (first major release), with the IRI "**https://w3id.org/example/1.0.0**", which we would represent in the ontology as follows:

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    owl:versionIRI <https://w3id.org/example/1.0.0> ;
    owl:versionInfo "1.0.0" .
```

As shown in the example, the version IRI is independent from the URI of the ontology ("https://w3id.org/example#"). It is **discouraged** to include version numbers as part of the ontology URI, as it would deeply affect interoperating with its instances. For example, consider we had used "https://w3id.org/example-/1.0.0#" as the ontology URI, and we had populated a knowledge graph with two instances of "Researcher":

```
@prefix exo: <https://w3id.org/example/1.0.0#> .
@prefix ex-inst: <https://example.org/instance/> .
ex-inst:alice a exo:Researcher .
ex-inst:bob a exo:Researcher .
```

If we now released another version of the ontology (1.0.1), all the URIs of the class Researcher would change (highlighted in blue below):

```
@prefix exo2: <https://w3id.org/example/1.0.1#> .
@prefix ex-inst: <https://example.org/instance/> .
ex-inst:alice a exo2:Researcher .
ex-inst:bob a exo2:Researcher .
```

[11] https://semver.org/

This is an undesired behavior, because it makes **ex-inst:alice** and **ex-inst:bob** instances of two different classes (**exo:Researcher** and **exo2:Researcher**). Instead, we want all the instances of a class to be compatible across different ontology versions:

```
@prefix exo: <https://w3id.org/example#> .
@prefix ex-inst: <https://example.org/instance/> .
ex-inst:alice a exo:Researcher .
ex-inst:bob a exo:Researcher .
```

By following this convention, we can continue doing ontology releases with appropriate versioning, while keeping the classes and properties URIs consistent.

### 2.5 Using permanent URIs

When publishing an ontology on the Web, it is recommended to think about its long term sustainability, specifically if it gets widely adopted. For example, what will happen to the domain used for the namespace URI of an ontology after several years? (i.e., when the funding for the related research project is over). Likewise, if the ontology is hosted on a server in a university or company, what will happen if the server domain name changes; or if the person in charge of maintaing the ontology needs to move it to another institution?

Permanent URIs services are community driven initiatives designed to address these issues. The idea behind permanent URI services is simple: instead of minting a new URI for a resource, users may use these services to create a *proxy* URI which can then be redirected to wherever the target resource is stored at any point in time. That way, if the target resource is moved, users just have to update its location without changing its permanent URL. There are several open, free services to create permanent URLs on the Web, among which purl.org[12] (now hosted by the Internet Archive) and w3id[13] (created by the W3C Permanent Identifier Community Group and supported by several companies) are perhaps the most commonly used. We recommend using permanent URIs in ontologies in order to support their long term sustainability. In fact, our example ontology uses a w3id: **https://w3id.org/example**. Creating a w3id is as simple as forking a GitHub repository and following the instructions in the readme file.[14] An advantage of w3id versus purl is that you have control on how to redirect the ontology to its different serializations (an example is available in Section 4).

## 3 Generating Reusable Ontology Documentation

We refer to "ontology documentation" as the collection of documents and explanatory comments generated during the entire ontology building process [14].

[12] `https://archive.org/services/purl/`
[13] `https://w3id.org/`
[14] `https://github.com/perma-id/w3id.org`

Having a proper documentation is critical, as it provides context, accurate definitions and examples of the different concepts that are included in an ontology. In fact, an important part of the documentation is usually provided within the ontology itself through ontology metadata and natural language annotations. However, some of the documents may be external to the ontology, such as the ontology requirements document, other sources used during the knowledge acquisition phase, the conceptualization diagrams, examples of use, etc.

In this section we describe our recommended best practices to generate ontology metadata and human-readable documentation, including diagramming guidelines to show the relationships between classes in a visual manner.

### 3.1 Ontology Metadata

When creating an ontology, it is crucial to describe it with appropriate metadata for others to understand it correctly. For example, if some of the classes are ambiguously defined, other researchers may misinterpret their meaning when incorporating them into their work. We distinguish two main categories of metadata in an ontology: the metadata associated with the ontology itself and the metadata associated with its elements (classes, object properties, datatype properties and individuals).

The metadata associated with the ontology itself is important to provide an overview and identify an ontology, understand its usage conditions and understand its provenance. Table 1 shows our recommended and optional annotation properties for describing ontologies, along with candidate properties that can be reused from existing vocabularies and standards.[15] The *recommended* properties are *license* (critical for others to know how the ontology may be used; we recommend a CC-BY license); *creator*, *contributor*, *creation date* and *previous version* to track the provenance of the ontology and compare against earlier versions; *namespace URI* and *version IRI* to properly identify the ontology; and *namespace prefix*, *title* and *description* to provide a quick overview on what the ontology does and how to properly refer to it. Finally, a *citation* is recommended for letting other users know how to attribute the authors of the ontology. For the rest of the section, we will be using the following namespaces:[16]

```
rdfs    <http://www.w3.org/2000/01/rdf-schema#>
owl     <http://www.w3.org/2002/07/owl#>
bibo    <http://purl.org/ontology/bibo/>
foaf    <http://xmlns.com/foaf/0.1/>
dcterms <http://purl.org/dc/terms/>
vaem    <http://www.linkedmodel.org/schema/vaem>
vann    <http://purl.org/vocab/vann/>
sw      <http://www.w3.org/2003/06/sw-vocab-status/ns#>
```

[15] Other vocabularies (e.g., `https://schema.org`) are alternatives to the ones proposed. See `https://w3id.org/widoco/bestPractices` for additional suggestions.

[16] For reference, the TTL version of our example ontology is available at `https://dgarijo.github.io/example/release/1.0.1/ontology.ttl`

**Table 1.** Recommended and optional metadata for describing ontologies

| **Property name** | **Annotation Property** | **Rationale** | **Guideline** |
|---|---|---|---|
| License | dcterms:license | Usage conditions | Recommended |
| Creator | dcterms:creator | Provenance and attribution | Recommended |
| Contributor | dcterms:contributor | Provenance and attribution | Recommended |
| Creation date | dcterms:created | Provenance | Recommended |
| Previous version | owl:priorVersion | Provenance and comparison | Recommended |
| Namespace URI | vann:preferredNamespaceUri | Identifying the ontology | Recommended |
| Version IRI | owl:versionIRI | Versioning | Recommended |
| Namespace prefix | vann:preferredNamespacePrefix | Identifying the ontology | Recommended |
| Title | dcterms:title | Understanding | Recommended |
| Description | dcterms:description | Understanding | Recommended |
| Citation | dcterms:bibliographicCitation | Credit | Recommended |
| Abstract | dcterms:abstract | Additional information | Optional |
| See also | rdfs:seeAlso | Additional information | Optional |
| Status | sw:status | Maturity information | Optional |
| Backward compatibility | owl:backwardCompatibility | Version compatibility | Optional |
| Incompatibility | owl:incompatibleWith | Version compatibility | Optional |
| Modification date | dcterms:modified | Provenance and timeliness | Optional |
| Issued date | dcterms:issued | Provenance and timeliness | Optional |
| Source | dcterms:source | Provenance | Optional |
| Publisher | dcterms:published | Provenance | Optional |
| DOI | bibo:doi | Bibliographic information | Optional |
| Logo | foaf:logo | Identifying the ontology | Optional |
| Diagram | foaf:depiction | Visual documentation | Optional |

The *optional* properties included in Table 1 are not critical to identify or reuse a target ontology, but provide additional insight and ease its understanding. These properties include having an *abstract* and *see also* with an additional overview of the ontology and links to related resources; a *status* to describe its maturity (first draft, specification, etc.); information about the *backward compatibility* or other *incompatible* versions of the ontology; the *modification* and *issue* dates; the *original source* that led to the definition of the ontology (requirement document, use cases, etc.); information about the *publisher* organization supporting the ontology; the *DOI* identifying a publication about the ontology; and information about the *logo* and *diagrams* that can be used as a visual aid for the ontology.

Table 2 shows the recommended and optional metadata properties for classes, properties, data properties and individuals. Recommended metadata properties include a human-readable *label* to identify an ontology term (using as many languages as needed); and a *definition* for the ontology term that is as accurate as possible. Definitions should be clear and illustrative, as classes and property names may have different meanings to different researchers.

The rest of the properties in Table 2 are nice to have to improve the understanding of ontology terms. These include *examples* that illustrate how to use a term; its *status* (e.g., deprecated, under discussion, etc.); the *rationale* for including a term in the ontology (which may reflect consensus from a discussion); the *source* material that motivated the inclusion of the term; and vocabulary where the term *is defined* in case of being reused.

**Table 2.** Recommended and optional properties for describing ontology terms

| **Property name** | **Annotation Property** | **Rationale** | **Guideline** |
|---|---|---|---|
| Label | rdfs:label | Readibility | Recommended |
| Definition | rdfs:comment | Understanding | Recommended |
| Example | vann:example | Understanding | Optional |
| Status | sw:term_status | Understanding | Optional |
| Rationale | vaem:rationale | Understanding | Optional |
| Source | dcterms:source | Provenance | Optional |
| Defined by | rdfs:isDefinedBy | Provenance | Optional |

### 3.2 Creating a Human-Readable Documentation

Ontologies are usually designed in editors and then exported in formats (Turtle, RDF/XML, JSON-LD, etc.) that are difficult to navigate by humans. Researchers often address this issue by pointing to paper or report that explains the ontology, but papers usually describe a scientific contribution rather than the definitions of each ontology concept in detail. A better solution is to create a documentation for all the terms in the ontology. Since this can be a time-consuming task, the Semantic Web community has developed tools to help ontology documentation. Given an OWL file as input, these tools generate an HTML documentation from the metadata included in the ontology itself (by retrieving the annotation properties recommended in Section 3.1), creating sections for all classes, properties, data properties and named individuals. Popular tools for ontology documentation include WIDOCO [3] (an evolution of the Live OWL Documentation Environment [12] that includes automated visualization diagrams through the WebVowl tool [9]), Parrot [2] or OwlDoc[17] (integrated with the Protégé Ontology editor [11]).

We recommend creating an HTML documentation for ontologies, as it makes them easier for others to understand and navigate on the Web. The tools introduced above are a good starting point for documentation generation, but we recommend expanding their results with additional motivation and context for the target ontology, pointers to the requirements and rationale; and custom diagrams with examples that illustrate how to use ontologies in practice.

### 3.3 Ontology visualization

Graphical representations of ontologies help users understand their structure, relationships and usage. Since there is no standard convention for ontology diagrams, researchers have adopted several approaches, such as UML-alike diagrams in the SAREF ontologies;[18] semantic network oriented diagrams in the SSN Ontology;[19] or custom diagrams as in the W3C PROV Ontology.[20]

[17] `https://protegewiki.stanford.edu/wiki/OWLDoc`
[18] See the SAREF4AGRI extension `https://w3id.org/def/saref4agri`
[19] Semantic Sensor Network Ontology `https://www.w3.org/TR/vocab-ssn/`
[20] PROV-O: The PROV Ontology `http://www.w3.org/TR/prov-o/`

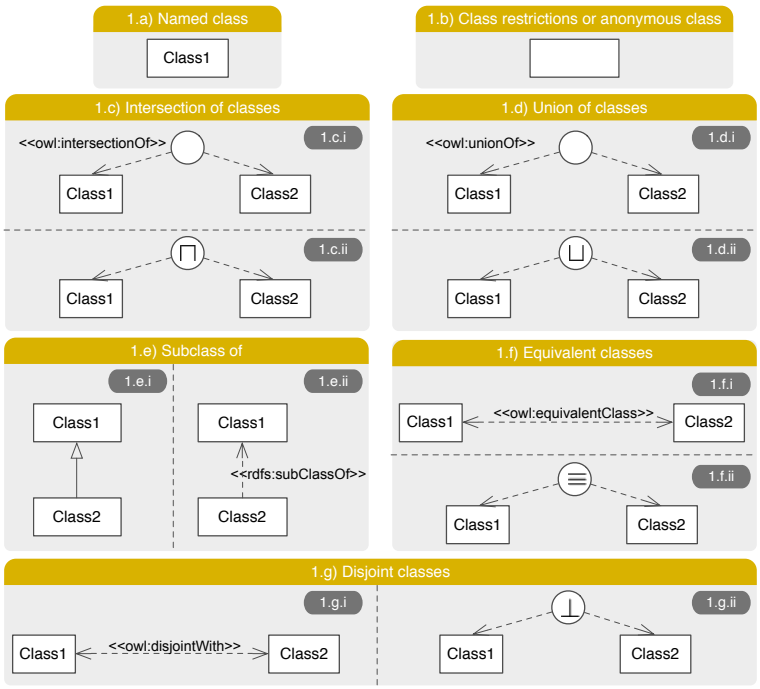


**Fig. 1.** Recommended notation for classes.

In the last years, conventions for ontology diagrams have been proposed (e.g., VOWL [10] and Graffoo[21]) but none have been standardized yet. In this section we suggest guidelines for generating ontology diagrams based on the UML_Ont profile proposed in [4].[22] The rationale for our recommendation is that UML is commonly used in software engineering, and it is familiar to software engineers.

Figure 1 depicts our proposed graphical representations for classes, class restrictions and class axioms. **Named classes** are represented by labelled boxes (1.a); while **class restrictions or anonymous classes** are represented by empty boxes (1.b). **Intersection class** descriptions are represented either by using an empty circle with the `<<owl:intersectionOf>>` stereotype (1.c.i); or an icon including the symbol "⊓" (1.c.ii). Similarly, union class descriptions may use an empty circle with the `<<owl:unionOf>>` stereotype (1.d.i); or an icon including the symbol "⊔"(1.d.ii).

**Subclasses** are represented using the generalization arrow used in UML (1.e.i) or with a dependency arrow with the `<<rdfs:subClassOf>>` stereotype (1.e.ii); and **equivalent classes** are represented with double-sided UML depen-

[21] Specification available at `https://essepuntato.it/graffoo/specification`

[22] The original UML_Ont profile uses custom labels and dependencies to cover OWL 1 constructs. Here labels are mapped to the OWL and RDF(S) constructs.

dency arrows with the <<owl:equivalentClass>> (1.f.i) stereotype or by a circle including the symbol "≡" (1.f.ii). Lastly, **disjoint classes** may be represented with double-sided UML dependency arrows, using the <<owl:disjointWith>> stereotype (1.g.i) or with a circle including the symbol "⊥" (1.g.ii).

Figure 2 illustrates guidelines on how to represent **object properties**. When the domain or range are not known (2.a), properties can be represented with dotted arrows (2.a.i); or with a diamond with the <<owl:ObjectProperty>> stereotype (2.a.ii). **Subproperties** may be represented with the UML dependency arrow with the <<owl:subPropertyOf>> stereotype linking the arrows that represent the involved object properties (2.b.i) or with an UML dependency unidirectional arrow with the <<owl:subPropertyOf>> stereotype linking the diamonds that represent the involved object properties (2.b.ii). When **domain** and **range** are known, properties can be represented with a solid line between source and target classes (2.c.i) or with a labelled diamond accompanied by dotted arrows labelled with <<rdfs:domain>> and <<rdfs:range>> respectively (2.c.ii). **Equivalent** (2.d) and **inverse** object properties (2.e) can be represented by using a bidirectional arrow with the <<owl:equivalentProperty>> and <<owl:inverseOf>> stereotypes between the lines (2.d.i and 2.e.i) or using diamond shapes (2.d.ii and 2.e.ii). Lastly, **functional** (2.f), **transitive** (2.g) and **symmetric** (2.h) object properties can be represented using a shared notation: either by adding the first initial of the property type (F, T or S) to the object property label attached to the arrow that represents the object property; or by a labelled diamond, which represents the object property itself, including the corresponding stereotype (e.g., <<owl:FunctionalProperty>>).

Figure 3 shows how to represent **datatype properties**. When the domain or range are not known (3.a) datatype properties may be represented as labelled dashed boxes attached to boxes representing classes (3.a.i); or as a diamond with <<owl:DatatypeProperty>>. **Subproperties** for datatypes may be represented with a UML dependency arrow with the <<owl:subPropertyOf>> stereotype linking the diamonds that represent the involved datatype properties (3.b). When the **domain** and/or **range** are known, the box representing the datatype property may be depicted with a solid line indicating that the domain of the datatype property is the attached class and the range may be included following the character ":" after the datatype label (3.c.i). Alternatively, a labelled diamond may be used accompanied by dotted arrows labelled with the <<rdfs:domain>> and <<rdfs:range>> stereotypes respectively (3.c.ii). **Equivalent datatype properties** may be represented by a UML dependency bidirectional arrow with the <<owl:EquivalentProperty>> stereotype linking the diamonds that represent the datatype properties (3.d). **Functional datatype properties** may be represented by adding "(F)" to the datatype property label (3.e.i) or by a labelled diamond including the <<owl:FunctionalProperty>> stereotype (3.e.ii).

Finally, Figure 4 proposes how to represent individuals and class membership. **Individuals** or instances may be represented by labelled boxes with underlined names or identifiers (4.a). Class membership may be represented with labelled

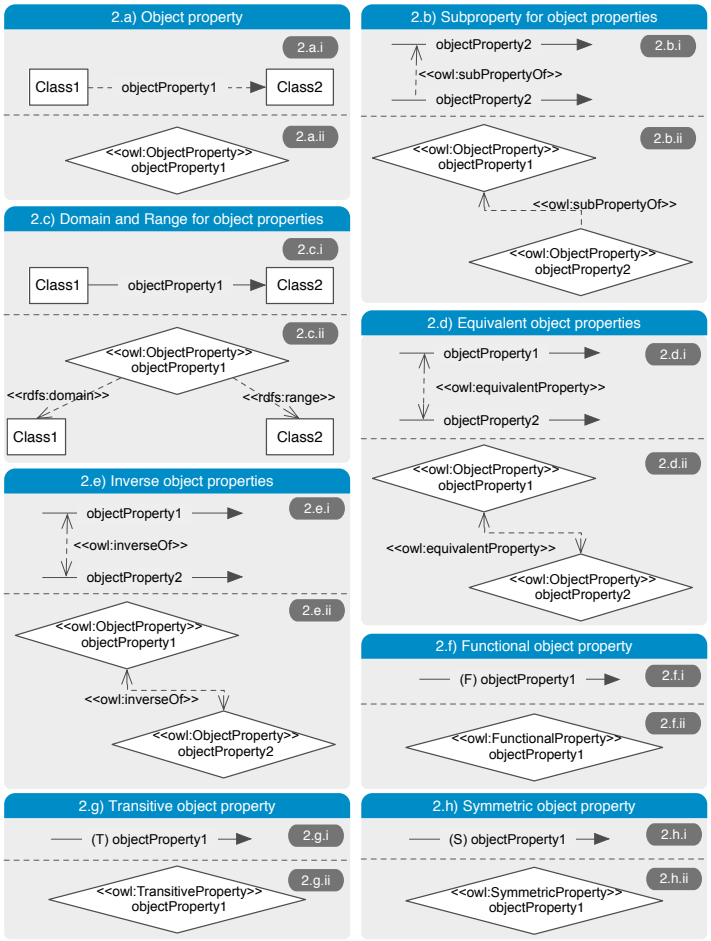


**Fig. 2.** Recommended notation for object properties.

boxes containing the individual name followed by the character ":" and the class name, all underlined (4.b.i); by linking the individual box with the class using a unidirectional UML dependency arrow with the stereotype `<<rdf:type>>` (4.b.ii); with a dashed line with a solid arrow (4.b.iii) or with a underlined labelled box for the individual attached to the class (4.b.iv).

## 4 Ontology Publication on the Web

Once an ontology is fully implemented and documented, it is time to make it accessible and findable in the Web. In this section we briefly describe the best practices to perform content negotiation to serve a target ontology in multiple formats (Section 4.1) and registries for making ontologies findable (Section 4.2).

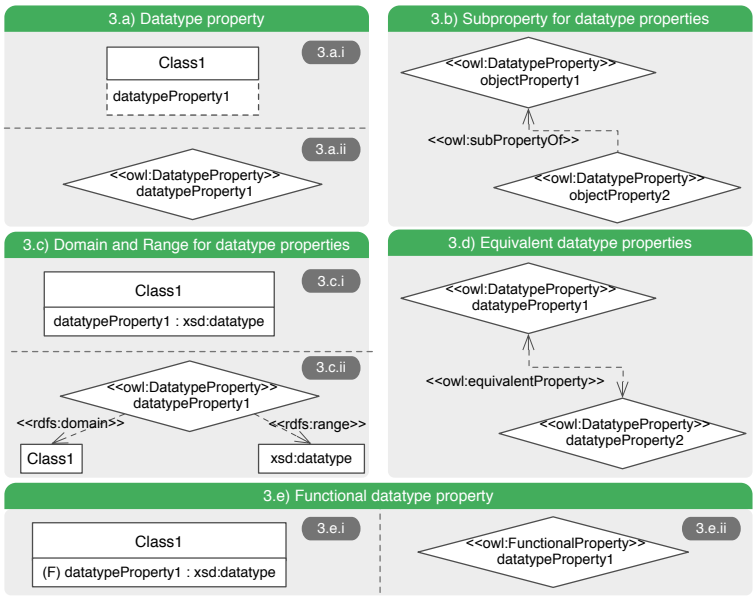


**Fig. 3.** Recommended notation for datatype properties.

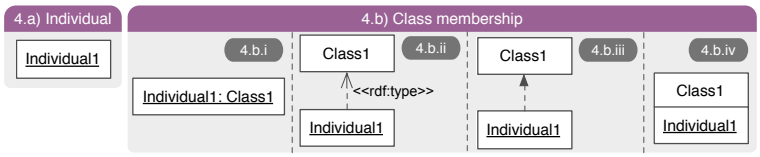


**Fig. 4.** Recommended notation for individuals.

### 4.1 Ontology Accessibility in Multiple Interoperable Formats

Ontologies should be made available in both human and machine readable manner using a single identifier: the ontology URI. This way we can make any ontology resolve to its HTML documentation when accessed by a user in a browser; and resolve to a standard RDF serialization when importing it in an ontology editor. In order to distinguish the target resource to serve (HTML or RDF serialization), we must implement a *303 See Other redirect*,[23] a common practice in the Semantic Web community for doing *content negotiation* over URIs. This type of redirect indicates the location of a target resource in the server based on the received request, but has to be appropriately configured on the server where we are hosting the ontology. Fortunately, there are W3C best practices (recipes) on how to configure an Apache HTTP server -a commonly used type

[23] `https://tools.ietf.org/html/rfc7231#page-57`

of server for serving files- for hash ended and slash ended ontologies.[24] Here we expand these practices with our example ontology to illustrate: 1) How to support multiple serializations of an ontology (HTML and Turtle); 2) how to support version redirection, as we would like all the versions of the ontology to be appropriately available, not only the latest; 3) How to specify if a serialization is not supported (for example, requests for JSON-LD will return a 406 non-acceptable code, rather than an RDF/XML serialization); and 4) how to implement a default response in case the user agent doing the request does not specify a target in the request (by default we return Turtle). The *htaccess* file to be placed in the server (in an **"/example"** folder) would look as follows:

```
# Turn off MultiViews (Apache-specific command)
Options -MultiViews

# Directive to specify supported types besides html and xml
# Standard RDF serialization formats include Turtle, RDF/XML, N-Triples and JSON-LD
AddType text/turtle .ttl
RewriteEngine on

# Rewrite rule for accessing the latest version.
RewriteCond %{HTTP_ACCEPT} !application/rdf\+xml.*(text/html|application/xhtml\+xml)
RewriteCond %{HTTP_ACCEPT} text/html [OR]
RewriteCond %{HTTP_ACCEPT} application/xhtml\+xml [OR]
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/.*
RewriteRule ^$ https://dgarijo.github.io/example/release/1.0.1/index-en.html [R=303,L]

# Rewrite rule to serve the Turtle serialization from the vocabulary URI (latest version)
RewriteCond %{HTTP_ACCEPT} text/turtle [OR]
RewriteCond %{HTTP_ACCEPT} text/\* [OR]
RewriteCond %{HTTP_ACCEPT} \*/turtle
RewriteRule ^$ https://dgarijo.github.io/example/release/1.0.1/ontology.ttl [R=303,L]

# Rewrite rules for retrieving a particular version (any version).
RewriteCond %{HTTP_ACCEPT} !application/rdf\+xml.*(text/html|application/xhtml\+xml)
RewriteCond %{HTTP_ACCEPT} text/html [OR]
RewriteCond %{HTTP_ACCEPT} application/xhtml\+xml [OR]
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/.*
RewriteRule ^(.+)$ https://dgarijo.github.io/example/release/$1/index-en.html [R=303,L]

# Rewrite rule to serve Turtle serialization of a particular version (any version)
RewriteCond %{HTTP_ACCEPT} text/turtle [OR]
RewriteCond %{HTTP_ACCEPT} text/\* [OR]
RewriteCond %{HTTP_ACCEPT} \*/turtle
RewriteRule ^(.+)$ https://dgarijo.github.io/example/release/$1/ontology.ttl [R=303,L]

# Rewrite rule for other non accepted formats
RewriteCond %{HTTP_ACCEPT} .+
RewriteRule ^(.*)$ https://dgarijo.github.io/example/release/1.0.1/406.html [R=406,L]

# Rewrite rule to serve the Turtle content from the vocabulary URI by default
RewriteRule ^$https://dgarijo.github.io/example/release/1.0.1/ontology.ttl [R=303,L]
```

In order to test the redirection, the easiest way is just to paste the URI of the ontology in Protégé or in your browser and check that both load the right serialization. Another possibility is to use a *curl* command,[25] e.g., to retrieve the Turtle serialization of our example ontology:

[24] http://www.w3.org/TR/swbp-vocab-pub/

[25] https://curl.haxx.se/

```
curl -sH "Accept: text/turtle" -L https://w3id.org/example#
```

If we need to access a particular ontology version, we can use its version IRI. For example, the documentation of the first version of our example ontology can be accessed with the following command:

```
curl -sH "Accept: text/html" -L https://w3id.org/example/1.0.0
```

### 4.2 Making an Ontology Findable on the Web

Once an ontology is published, the next step is to ensure it can be easily found by others. There are three main activities that can help the visibility of an ontology:

1. **Register the namespace prefix** using prefix.cc,[26] a crowdsourced registry where users can vote the most popular URI for a given prefix.
2. **Register the ontology**: There are a number of existing metadata registries that can be used for browsing existing ontologies [15][17]. Our recommendation is to look first for domain-specific registries (e.g., Bioportal [17] in the biomedical domain, Agroportal [7] in Agronomy, etc.) commonly used by the target community of interest. When domain-specific registries do not exist, we suggest registering the ontology in a domain-generic metadata registry, such as Linked Open Vocabularies [15] (which has a manual curation process to ensure that minimum metadata is provided) or FAIRsharing.[27]
3. **In-document annotations** to help crawlers understand the metadata of the ontology when publishing it on the Web. These annotations can be added in your documentation through JSON-LD snippets,[28] as shown below:

```
<!-- Annotations for the example ontology -->
<script type="application/ld+json">{
   "@context":"http://schema.org",
   "@type":"WebPage",
   "url":"https://w3id.org/example",
   "name":"The example ontology",
   "datePublished":"5-2-2020",
   "version":"1.0.1",
   "license":"http://creativecommons.org/licenses/by/2.0/",
   "author":[{"@type":"Person","name":"Daniel Garijo"},
             {"@type":"Person","name":"María Poveda"}],
}</script>
```

## 5 Ontology Documentation and Publication Frameworks

Semantic Web researchers and practitioners have developed methods and tools for easing ontology engineering, development, publication and exploitation. A

[26] http://prefix.cc
[27] https://fairsharing.org/standards/
[28] https://www.w3.org/TR/json-ld11/

number these tools have already been mentioned in the corresponding sections of this chapter, however, they are not always integrated as part of an end-to-end framework, and researchers have to use them separately (e.g., generate the documentation of an ontology with WIDOCO, create a visualization with WebVOWL and publish it online using GitHub).

More recently, frameworks inspired by the continuous integration practices in software engineering have arisen to support ontology engineering activities. These frameworks offer end-to-end solutions that support ontology engineers documenting, visualizing, testing and publishing their ontologies; and we recommend them as an entry point to adopt some of the practices described in this chapter. One example is OnToology [1], a web application[29] that orchestrates ontology documentation, evaluation and publication on the Web with permanent URLs. Another similar approach is VoCol [5], which provides ontology engineers with feedback on syntax and other errors and gives access to a human-readable presentation of a target ontology. Finally, PoolParty [13] is a commercial solution that also includes publication of thesauri, taxonomies and ontology management among other features.

## 6 Conclusions

In this chapter we have described implementation guidelines and recommendations for making ontologies findable (through metadata registries and annotations); accessible (through good practices in URI design and content negotiation), interoperable (showing how to serve ontologies in different standard serializations) and reusable (by describing the metadata and diagram guidelines needed for proper understanding) on the Web while following the Linked Data principles. A distinct feature of our guidelines is that we have illustrated how to carry out our recommendations with an example ontology and pointers to usable tools developed by the Semantic Web community in the last decade. Our recommendations reflect years of experience in ontology engineering and also summarize community discussions for ontology design and publication. Hence, we believe these guidelines are a comprehensive starting point for ontology engineers who aim to make their ontologies FAIR and available on the Web.

## References

1. Alobaid, A., Garijo, D., Poveda-Villalón, M., Santana-Perez, I., Fernández-Izquierdo, A., Corcho, O.: Automating ontology engineering support activities with OnToology. Journal of Web Semantics (2018)
2. Alonso, C.T., Berrueta, D., Polo, L., Fernández, S.: Current practices and perspectives for metadata on web ontologies and rules. International Journal of Metadata, Semantics and Ontologies **7**(2), 93 (2012)
3. Garijo, D.: WIDOCO: A Wizard for Documenting Ontologies. In: The Semantic Web – ISWC 2017, vol. 10588, pp. 94–102. Cham (2017)

[29] `http://ontoology.linkeddata.es/`

4. Haase, P., Brockmans, S., Palma, R., Euzenat, J., d'Aquin, M.: D1.1.2 updated version of the networked ontology model. Tech. rep., Universität Karlsruhe (2009), NeOn Project. http://www. neon-project. org
5. Halilaj, L., Petersen, N., Grangel-González, I., Lange, C., Auer, S., Coskun, G., Lohmann, S.: Vocol: An integrated environment to support version-controlled vocabulary development. In: 20th International Conference on Knowledge Engineering and Knowledge Management - EKAW 2016, Bologna, Italy. Lecture Notes in Computer Science, vol. 10024, pp. 303–319 (2016)
6. Janowicz, K., Hitzler, P., Adams, B., Kolas, D., Vardeman II, C.: Five stars of Linked Data vocabulary use. Semantic Web **5**(3), 173–176 (2014)
7. Jonquet, C., Toulet, A., Arnaud, E., Aubin, S., Yeumo, E.D., Emonet, V., Graybeal, J., Laporte, M.A., Musen, M.A., Pesce, V., Larmande, P.: Agroportal: A vocabulary and ontology repository for agronomy. Computers and Electronics in Agriculture **144**, 126 – 143 (2018)
8. Le Franc, Y., Parland-von Essen, J., Bonino, L., Lehväslaiho, H., Coen, G., Staiger, C.: D2.2 FAIR Semantics: First recommendations (Mar 2020), `https://doi.org/10.5281/zenodo.3707985`
9. Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: Web-based visualization of ontologies. In: Proceedings of EKAW 2014 Satellite Events. LNAI, vol. 8982, pp. 154–158. Springer (2015)
10. Lohmann, S., Negru, S., Haag, F., Ertl, T.: VOWL 2: User-oriented visualization of ontologies. In: Proceedings of the 19th International Conference on Knowledge Engineering and Knowledge Management (EKAW '14). LNAI, vol. 8876, pp. 266–281. Springer (2014)
11. Musen, M.A.: The Protégé project: a look back and a look forward. AI Matters **1**(4), 4–12 (Jun 2015)
12. Peroni, S., Shotton, D., Vitali, F.: The Live OWL Documentation Environment: A Tool for the Automatic Generation of Ontology Documentation. In: Knowledge Engineering and Knowledge Management, vol. 7603, pp. 398–412. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
13. Schandl, T., Blumauer, A.: Poolparty: Skos thesaurus management utilizing linked data. In: Extended Semantic Web Conference. pp. 421–425. Springer (2010)
14. Suárez-Figueroa, M.C.: NeOn Methodology for building ontology networks: specification, scheduling and reuse. Ph.D. thesis, Facultad de Informatica, Universidad Politécnica de Madrid (2010)
15. Vandenbussche, P.Y., Atemezing, G.A., Poveda-Villalón, M., Vatant, B.: Linked Open Vocabularies (LOV): A gateway to reusable semantic vocabularies on the Web. Semantic Web **8**(3), 437–452 (Jan 2017)
16. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM **57**(10), 78–85 (Sep 2014)
17. Whetzel, P.L., Noy, N.F., Shah, N.H., Alexander, P.R., Nyulas, C., Tudorache, T., Musen, M.A.: BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications. Nucleic Acids Research **39**(suppl), W541–W545 (Jul 2011)
18. Wilkinson, M.D., Dumontier, M., Aalbersberg, I.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., et.al.: The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data **3**, 160018 (Mar 2016)