About versioning ontologies or any digital objects with clear semantics

Clement Jonquet, María Poveda-Villalón

About versioning ontologies or any digital objects with clear semantics

Clement Jonquet[1,2][0000-0002-2404-1582] María Poveda-Villalón[3][0000-0003-3587-0367]

[1] LIRMM, University of Montpellier & CNRS, France
[2] MISTEA, University of Montpellier, INRAE & Institut Agro, France
[3] Ontology Engineering Group, Universidad Politécnica de Madrid, Spain
clement.jonquet@inrae.fr
mpoveda@fi.upm.es

Abstract. The article discusses the process of versioning for ontologies and semantic artefacts developed using Semantic Web technologies. We describe methods for encoding versioning and other relevant information in metadata properties and we illustrate with examples from the MOD2.0 specification. Building on our experiences with the AgroPortal ontology repository and the Linked Open Vocabularies, we raise several questions, such as which metadata properties to use, how metadata values should be coordinated, and what stay the same over versions, and what should change. We propose recommendations for better versioning ontologies with clear semantics –identifiers, descriptions, status, dates, links– and suggests that the recommendations can be generalized to any digital objects that need to be versioned and semantically described.

Keywords: versioning, ontologies, semantic artefacts, metadata.

1 Introduction

Versioning is the process to produce a new version of a digital object. In our case, we focus on one type of digital object: ontologies and more largely semantic artefacts –a broader term to include ontologies, terminologies, taxonomies, thesauri, vocabularies, metadata schemas and metadata standards– developed with semantic web technologies (e.g., OWL, RDF-S or SKOS) and that we can describe with metadata properties –typically on the `owl:Ontology` or `skos:ConceptScheme` objects.

In software development, versioning is mainly viewed as the process of "*assigning either unique version names or unique version numbers to unique states of computer software*."[1] The numbering can even encode a certain "semantics" with the number changes meaning something[2] e.g., going from v2.2 to v2.3 corresponds to a more significant change than going from v2.2 to 2.2.1. In this article, we are not going to focus on such guidelines or practice but rather on the methods to encode such a versioning information –and much more– in relevant metadata properties. For instance, the versioning information is typically encoded in a semantic artefact with a property coming from the myriad of existing metadata vocabularies such as `owl:versionInfo` or `pav:version` or `schema:version` or `omv:version` `oboinowl:hasVersion`. In fact, we are not going to discuss neither which metadata property to use but focus on which value to give them and how to consistently edit them. We will

[1] https://en.wikipedia.org/wiki/Software_versioning
[2] https://semver.org/spec/v2.0.0.html

illustrate our speech with the default suggested properties in the MOD2.0 specification [1] (https://github.com/FAIR-IMPACT/MOD) –a proposed standard to describe ontologies and semantic artefact metadata– which includes the relevant "mappings" to other metadata properties. For instance, in MOD 2.0, the suggested property to encode the version information is `owl:versionInfo`.

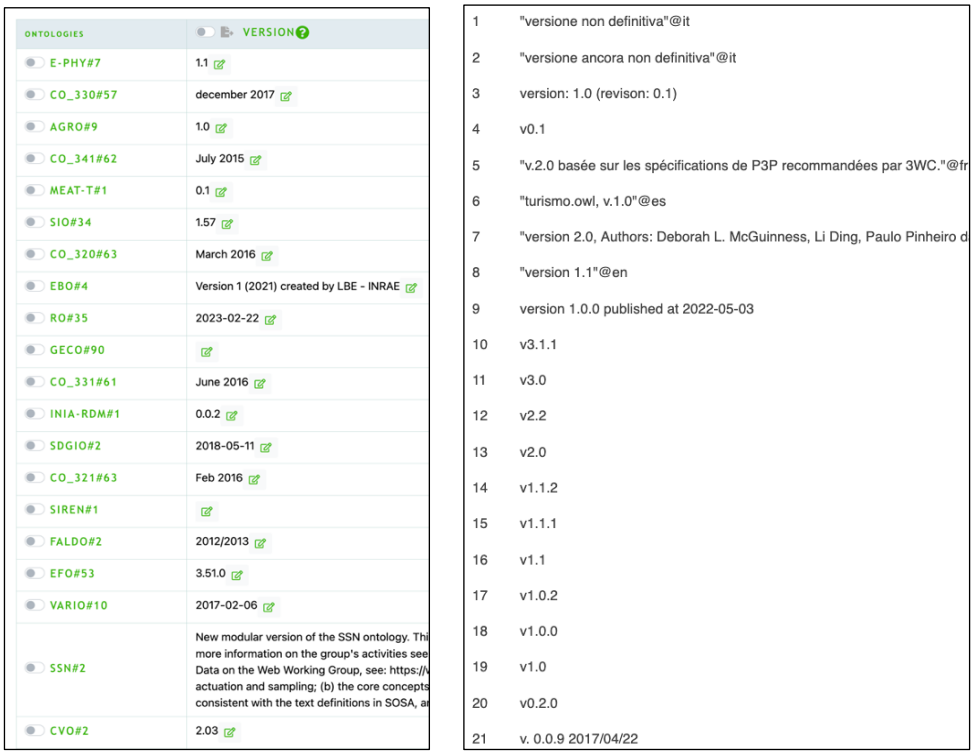
In fact, multiple metadata properties need to be filled in to properly encode the process of versioning; we will see there are a bunch of metadata properties that are linked to or affected by versioning such as identifiers, status, download links, relations to earlier versions, dates. And of course, these metadata properties should evolve logically when a new version of an artefact is produced. Then, multiple questions can be raised when thinking about versioning and metadata:

- Is it so different to encode the version information with a number (v1.4.2) than encoding it with a date (v2022-12-22)?
- Which dates are supposed to changes and which dates are supposed to stay the same when a new version of a resource is produced?
- What is the difference between an URI and a versioned URI? How does it impact external identifiers such as a DOI?
- How to encode a description or some notes specific to an ontology version distinguishing from a description "stable" over the versions?
- When a new ontology version is produced are the other ones deprecated or retired? Can an ontology be deprecated while keeping a production status?
- How metadata values need to be coordinated? So that, for example, the deprecated status and date of validity are coherent.
- How can metadata property values be "automatically" assigned when a new version is produced?

The bad news is that it is the responsibility of the ontology developer to pay attention to the completeness, accuracy and coherence of the metadata values. The good news is that it can be very well automated.

Building AgroPortal [2] (https://agroportal.lirmm.fr), a vocabulary and ontology repository (aka. semantic artefact catalogue) for agri-food, and managing the Linked Open Vocabularies (LOV) server [3], a widely used repository of semantic web vocabularies and ontologies, we constantly face situations where ontology developers need guidelines or recommendations on how to manage versioning. This situation has significantly increased when we developed an enriched and harmonized metadata model [4] and later used it to automatically evaluate the level of FAIRness of semantic resources hosted in AgroPortal [5, 6]. Fig. 1 to Fig. 4 illustrate some problematic situations which occur(ed) in AgroPortal or LOV. For example, different uses of the property version info (Fig. 1), same version information for different release dates (Fig. 2), not providing modifications date information (Fig. 3 and 4) and not providing modification dates updates for newer versions (Fig. 4).

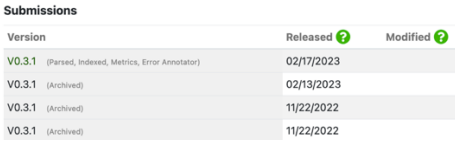
In this paper, we propose recommendations to better versioning ontologies with clear semantics. Indeed, despite uncomplete elements in recent FAIRness assessment related papers [6–9], we have not found a complete set of guidelines such as for instance for service-oriented systems [10]. Looking at it *at-posteriori*, our analysis and recommendations are not necessarily limited to ontologies or semantic artefacts but can be generalized to any digital objects (dataset, publication, software, workflow, etc.) that need to be versioned and be semantically described as discussed in the last Section.



ONTOLOGIES	VERSION
E-PHY#7	1.1
CO_330#57	december 2017
AGRO#9	1.0
CO_341#62	July 2015
MEAT-T#1	0.1
SIO#34	1.57
CO_320#63	March 2016
EBO#4	Version 1 (2021) created by LBE - INRAE
RO#35	2023-02-22
GECO#90	
CO_331#61	June 2016
INIA-RDM#1	0.0.2
SDGIO#2	2018-05-11
CO_321#63	Feb 2016
SIREN#1	
FALDO#2	2012/2013
EFO#53	3.51.0
VARIO#10	2017-02-06
SSN#2	New modular version of the SSN ontology. Thi more information on the group's activities see Data on the Web Working Group, see: https:// actuation and sampling; (b) the core concepts consistent with the text definitions in SOSA, a
CVO#2	2.03

1	"versione non definitiva"@it
2	"versione ancora non definitiva"@it
3	version: 1.0 (revison: 0.1)
4	v0.1
5	"v.2.0 basée sur les spécifications de P3P recommandées par 3WC."@fr
6	"turismo.owl, v.1.0"@es
7	"version 2.0, Authors: Deborah L. McGuinness, Li Ding, Paulo Pinheiro d
8	"version 1.1"@en
9	version 1.0.0 published at 2022-05-03
10	v3.1.1
11	v3.0
12	v2.2
13	v2.0
14	v1.1.2
15	v1.1.1
16	v1.1
17	v1.0.2
18	v1.0.0
19	v1.0
20	v0.2.0
21	v. 0.0.9 2017/04/22

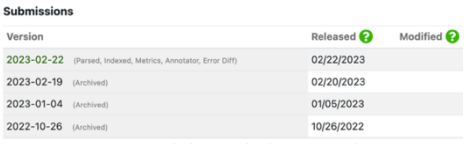
Fig 1. Variety of values for the version information in AgroPortal (screenshot from the administration interface) (left) and LOV SPARQL endpoint (right).



Submissions

Version	Released	Modified
V0.3.1 (Parsed, Indexed, Metrics, Error Annotator)	02/17/2023	
V0.3.1 (Archived)	02/13/2023	
V0.3.1 (Archived)	11/22/2022	
V0.3.1 (Archived)	11/22/2022	

Fig 2. Case of the Food Ontology (FOODON) which does not include any versioning information inside the source file nor modification dates, and that has not been curated (yet).



Submissions

Version	Released	Modified
2023-02-22 (Parsed, Indexed, Metrics, Annotator, Error Diff)	02/22/2023	
2023-02-19 (Archived)	02/20/2023	
2023-01-04 (Archived)	01/05/2023	
2022-10-26 (Archived)	10/26/2022	

Fig 3. Case of the Relation Ontology (RO) which uses a date as a versioning information but does not provide the modification date in the appropriate field.

2 Related work

In the literature, ontology versioning refers to the process of managing and tracking changes –and their effects– made to an ontology [11]. As ontologies evolve over time, it is essential to keep track of their changes to maintain compatibility with existing applications and data. Ontology versioning has always been identified as a key element of ontology management [12] and the need to track metadata related to versioning (our subject of interest here) was identified in early systems such as OntoView [13]. Multiple research studies looked into the type, frequency and representation of changes in ontologies such as [14] or [15].

Submissions

Version	Released	Modified	Uploaded
2022-09 (Parsed, Indexed, Metrics, Annotator, Error Diff)	03/02/2023	02/28/2022	03/02/2023
2022-09 (Archived)	02/02/2023	02/28/2022	02/02/2023
2022-09 (Archived)	01/10/2023	02/28/2022	01/10/2023
2022-09 (Archived)	12/01/2022	02/28/2022	12/01/2022
2022-09 (Archived)	11/03/2022	02/28/2022	11/03/2022
2022-09 (Archived)	10/13/2022	02/28/2022	10/13/2022
2022-09 (Archived)	09/02/2022	09/02/2022	09/02/2022
2022-07 (Archived)	01/01/1980	07/01/2022	07/01/2022
2022-06 (Archived)	01/01/1980	06/01/2022	06/01/2022
2022-05 (Archived)	01/01/1980	05/01/2022	05/20/2022

Fig 4. Case of AGROVOC which does not include versioning information and dates in the source file and for which curation was made by AgroPortal's team until version 2022-09.

In this last paper, the authors propose an ontology versioning framework capable to maintain the relationship among different version of ontology explicit by representing changes at "term level." The "meta level" is also identified and examples of metadata properties (from classic metadata vocabularies) are given, but without policy on how to fil them and make them coherent. Recently, we can also cite the Knowledge Graph Change Language (KGCL)[3] developed to describing change operations for ontologies or knowledge graphs.

In [16], the authors present the idea of an HTTP-based versioning mechanism that aims to provide a simple and efficient way to manage changes in Linked Data or for Web resources in general. Here again the authors identified metadata properties that should be used to properly track versioning information and the relation between these properties (our subject of interest here) appear e.g., of previous/next versions coherent information.

Another set of work related to versioning and ontologies is reported in [17], where the authors review the different approaches to version control (similar to what's done for software) for RDF data. However, this is not related to how to encode and what to encode at the ontology metadata level for better versioning information.

3 Recommendations to better versioning ontologies

3.1 How to use URIs, versioned URIs and external identifiers?

Ontologies, as any resource in the semantic web shall be assigned a Uniform Resource Identifier (URI) or Internationalized Resource Identifier (IRI). IRIs are minted and under the responsibility of the organization creating them; they are usually globally unique, but their persistent and resolvable characteristics are not guaranteed. One can follow good practices for URIs e.g., W3C best practices[4] and/or consider using PURLs or W3IDs. The reader can also check Section 2 of Garijo & Poveda's recommendation [18]. Typically, the IRI of an ontology shall not be versioned i.e., it must not contain any version information (number, date, text) but shall stay the same through versions. The idea is that the IRI, as an identifier,

[3] https://incatools.github.io/kgcl
[4] https://www.w3.org/2011/gld/wiki/223_Best_Practices_URI_Construction

shall not change so that users do not need to update their reference to the re-used ontology (or each object inside the ontology) each time a new version is available.

In addition, a property `owl:versionIRI` can be used to store the specific IRI of the version. The OWL specification says:[5] "*Each ontology may have an ontology IRI, which is used to identify an ontology. If an ontology has an ontology IRI, the ontology may additionally have a version IRI, which is used to identify the version of the ontology*." The difference comes when the IRIs are dereferenced: the `versionIRI` shall always resolve to a specific source file corresponding to the specific ontology version or another relevant page (depending on content negotiation);[6] whereas the ontology IRI shall redirect to the latest `owl:versionIRI` in order to resolve to the latest ontology source file available. It means that each time a new ontology version is created, developers must change the IRI resolving mechanism (e.g., simple HTTP redirects) to update the target of the ontology IRI as illustrated in Fig. 5.

In some cases, ontologies can be assigned an additional Permanent IDentifier (PID) by an external organization such as a DOI. When it comes to the external identifier, it is up to the ontology developer to self-inform when it is required to create a new identifier for a new version for an object. For example, DataCite recommends creating a new DOI "if there is major change to the content being shared".[7] AgroPortal has taken the convention that external identifiers shall always be different than URIs to reflect the principle that this "second identifier" is assigned by an external body, independently of the developing organization. Because the ontology IRI is not a "property' but the RDF resource identifier, in MOD, another metadata property (`mod:URI`) was explicitly created to manipulate the URI value as any other metadata property and the property `dct:identifier` is used to encode another "external" identifier. The following statements are therefore recommended to declare identifiers:[8]

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    mod:URI "https://w3id.org/example" ;
    owl:versionIRI <https://w3id.org/example/1.0> ;
    dct:identifier "10.15454/1.4656E12" .
```

And when a new version is produced:

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    mod:URI "https://w3id.org/example" ;
    owl:versionIRI <https://w3id.org/example/2.0> ;
    dct:identifier "10.15454/1.4656E12" .
```

[5] https://www.w3.org/TR/owl2-syntax/#Ontology_IRI_and_Version_IRI

[6] Section 3.6 presents the `dcat:accessURL` and `dcat:downloadURL` that can be used to store the target URLs for resolving the ontology `versionIRI` with content negotiation.

[7] https://support.datacite.org/docs/versioning

[8] Examples are continued from Garijo & Poveda's 2020 paper [18].

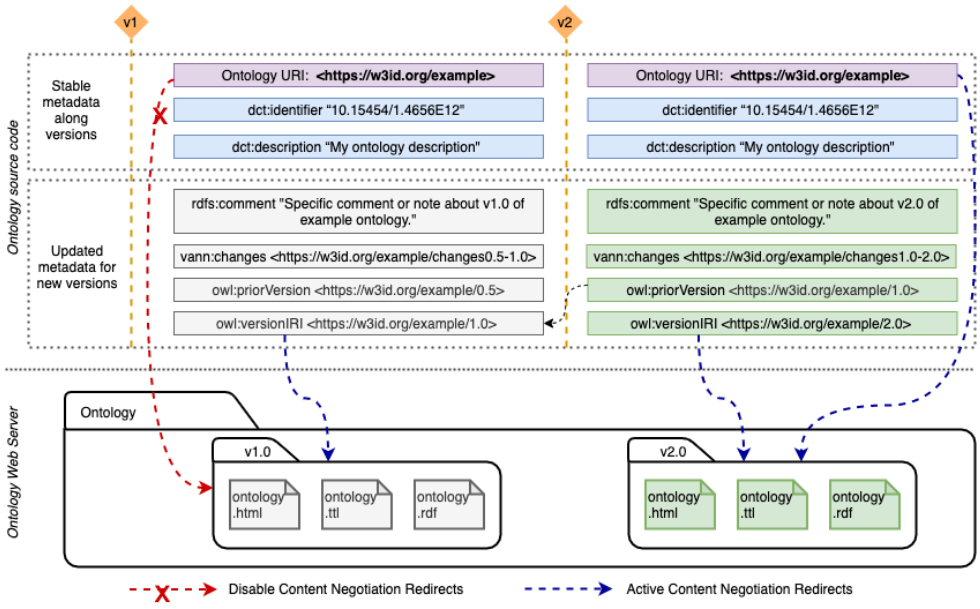


Fig. 5. Illustration of some metadata properties stability or update between two versions inside the source code. The figure also illustrates the target Web server files of URIs dereferencing.

3.2 How to describe the ontology, version information, changes between versions?

Providing a description, typically with `dct:description` is a good thing. It is usually a short paragraph of text which explains what is this ontology and why it was created, etc. The description typically "stays for long" and does not need to change at each new version of the ontology. However, describing a versioned ontology requires a bit more of information. It is appropriate to add an `owl:versionInfo` metadata property each time an ontology developer releases a new version so that anyone reusing the ontology knows exactly which version he/she is using. But this metadatum must only contain the version and no other information (dates, comments or creators as illustrated in Fig. 1). As Garijo & Poveda 2020, we do encourage to use *semantic versioning* (https://semver.org), rather than dates or miscellaneous strings or mixed of characters and numbers. Mostly because, in addition of carrying a semantics, a standard format like *semver* can also be parsed and automatically processed. Using the date as a version information is not recommended although possible, as there could be ambiguity on what the date exactly represent (see later section on dates). Plus, dates are hardly exploitable (see variety of date formats in Fig. 1) and hide if this is a major or minor release. Because of their importance, in AgroPortal (historically in BioPortal), both `dct:description` and `owl:versionInfo` are two mandatory properties for each ontology versions.

In addition, the properties `rdfs:comments` and `vann:changes` may be used to respectively describe something specific to a given version and document or refer to the list (possibly expressed in a formal language)[9] of changes in the new version (e.g., classes/properties added or removed, etc.). In AgroPortal, or with the Widoco documentation generation

[9] A recent initiative for that is KGCL: https://github.com/INCATools/kgcl

tool [19], the changes are automatically computed with the Bubastis Ontology diff tool[10] and the results can be use as the value of the `vann:changes` property.

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    dct:description "Global description of example ontology which
stays for long."@en ;
    owl:versionInfo "1.0" ;
    rdfs:comment "Specific comment or note about v1.0 of example on-
tology."@en .
```

And when a new version is produced:

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    dct:description "Global description of example ontology which
stays for long."@en ;
    owl:versionInfo"2.0" ;
    rdfs:comment "Specific comment or note about v2.0 of example on-
tology. "@en ;
    vann:changes "Description of changes or reference to a resource
that describes the changes between v1.0 and v2.0 of example ontol-
ogy"@en .
```

3.3 How to inform about the ontology status and what to do when an ontology becomes obsolete?

When describing a versioned ontology, it might be relevant to inform about the different production phases and clearly describe the status especially if it becomes obsolete or deprecated. In MOD, `mod:status`, a property (and its values) inherited and adopted from the Ontology Metadata Vocabulary (OMV) [20] indicates the different production phases (`alpha`, `beta`, `production`, `retired`) and the property `owl:deprecated` (boolean[11]) indicates if the ontology is deprecated. Using two properties (rather than one additional status=deprecated) allows to express the idea that an ontology can be deprecated – i.e., not maintained anymore, tolerated or supported but not recommended– but not necessarily retired – i.e., not supported any more, possibly even not available anymore at its original source.[12] However, these two properties are linked one another and should be used consistently inside a given ontology version and through multiple versions:

- For any version, if `status=retired` then `deprecated=true` but not the opposite.
- If the status of an ontology version is retired then the status of all the previous versions shall also certainly be retired. With the exception of a new version produced before a rollback to the previous one.
- If a new ontology version is created it must have `deprecated=false` or nothing. In some cases, all the previous version can then have `deprecated=true` (but not necessarily `retired=true`), for instance if the ontology developer does not recommend to use the previous version(s) at all. This might not be always the case, as in software

[10] https://github.com/EBISPOT/bubastis

[11] Even though the `owl:deprecated` annotation property's range is defined as `rdfs:Resource`, it is a common practice to provide boolean values when using the `owl:deprecated` predicate.

[12] Depreciation and retirement are two notions which meaning varies depending on the object concerned. We have not found a formal, standard or recognized definition when it comes to ontologies or semantic artefacts.

engineering, it might happen that a new version does not mean the old one is not maintained or supported anymore.

An example of situation with three successive versions of an ontology is:

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    owl:versionInfo "1.0" ;
    mod:status "retired" ;
    owl:deprecated "true"^^xsd:boolean .
```

Then:

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    owl:versionInfo "2.0" ;
    mod:status "production" ;
    owl:deprecated "true"^^xsd:boolean.
```

Then:

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    owl:versionInfo "3.0" ;
    mod:status "production" ;
    owl:deprecated "false"^^xsd:boolean.
```

3.4 How to encode dates appropriately?

When implementing the new metadata model for AgroPortal [4], the authors reviewed 15 metadata vocabularies and found out a strong overlap in all the vocabularies which more or less all redefine things that have already been described several times before, such as dates for which 25 properties are available. In the following, we propose to use three specific date properties for versioning to capture: (i) when an ontology has been originally created or released (`dct:created`), typically this date should be the same through ontology versions unless a major change occurred and the developer wants to make a time stamp e.g., typically with the Semver approach, a change of the first digit in the version; (ii) when an ontology has been modified (`dct:modified`), typically this date should be changing at each version; (iii) when an ontology version has/will become invalid (`dct:valid`), typically deprecated and/or replaced by another one. Here again, those three dates shall be coherent one another and with the status previously discussed:

- The modification date must always be after or equal to the creation date and usually before or equal to the validity date.
- The modification date of a new version must be after or equal to the modification and validity date of the previous version.
- For any version, if `deprecated=true` then validity date should be before or equal the current date i.e., it should capture when an ontology became deprecated.
- Reversely for any version, if a validity date is before the current date then `deprecated=true`.

The following code excerpts exemplify how to encode creation, modification and validity dates for three consecutive ontology versions according to the guidelines above-mentioned.

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    owl:versionInfo "1.0" ;
```

```
    owl:deprecated true ;
    dct:created "2020-01-01" ;
    dct:modified "2020-01-01" ;
    dct:valid "2020-12-12" .
```

Then:

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    owl:versionInfo "2.0" ;
    owl:deprecated true ;
    dct:created "2020-01-01" ;
    dct:modified "2021-01-01" ;
    dct:valid "2022-01-05" .
```

Then:

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    owl:versionInfo "3.0" ;
    owl:deprecated false ;
    dct:created "2020-01-01";
    dct:modified "2022-01-05" ;
    dct:valid "2024-12-31" .
```

In AgroPortal (in Fig. 2 to 4), `dct:created` is called 'Released' and `dct:modified` is called 'Modified'. Plus, AgroPortal uses another property to store the date when a new version (called 'submission' in the system) is uploaded to the portal (called "Uploaded" in the figures) and it is automatically generated. In MOD, such an information could be represented with `dct:dateSubmitted`. It also uses the date property `pav:curatedOn` to inform when an ontology has been curated or evaluated but this is not related to versioning.

3.5 How to encode the relations between multiples versions?

Properly defining the metadata of a research object also means formally encoding the relation between this object and others. This applies to semantic artefact too. The MOD metadata model does provide 20 properties to describe relations between ontologies (imports, specialization/generalization, alignment, similarity, etc.), a few of them are relevant when versioning an ontology. The OWL specification recommends to declare explicitly the link to the unique previous version of the ontology with the property `owl:priorVersion` –one can also declare backward compatibility or incompatibility– and Dublin Core offers the mechanism to link back to all the previous versions with the property `dct:hasVersion`.[13] For these properties, one must use the `owl:versionIRI` values; in our example this gives:

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    owl:versionIRI <https://w3id.org/example/2.0> ;
    owl:priorVersion <https://w3id.org/example/1.0> ;
    owl:backwardCompatibleWith <https://w3id.org/example/1.0> .
```

Then:

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    owl:versionIRI <https://w3id.org/example/3.0> ;
```

[13] Also available in the provenance vocabulary PAV[13] and recently being adopted by DCAT v3.

```
    owl:priorVersion <https://w3id.org/example/2.0> ;
    owl:backwardCompatibleWith <https://w3id.org/example/2.0> ;
    owl:incompatibleWith <https://w3id.org/example/1.0> ;
    dct:hasVersion <https://w3id.org/example/1.0> ,
                   <https://w3id.org/example/2.0> .
```

3.6 How to update related links when a new version is created?

Beyond IRIs that shall be resolvable/dereferenceable (but in reality, are often not), a semantic artefact can contain multiple links (i.e., URLs to specific web pages or web resource) to related information or the actual source file(s) corresponding to the semantic artefact. Some of them may change when versioning an ontology. Here, we follow the DCAT vocabulary approach which provides multiple metadata properties to: (i) link to a web landing page "that can be navigated in a Web browser" (`dcat:landingPage`), which provides general information, documentation, links, etc. for the given ontology and typically does not change through ontology versions; (ii) link to a web page or service where a specific ontology version can be accessed / browsed / queried / visualized (`dcat:accessURL`); (iii) link(s) to file(s) where a specific ontology version "distribution" can be downloaded in a specific format or language (`dcat:downloadURL`). Typically, `dcat:accessURL` and `dcat:downloadURL` maybe used to store within the ontology source file, the URLs used to dereference the `owl:versionIRI` with content negotiation.

```
<https://w3id.org/example> rdf:type owl:Ontology ;
    owl:versionInfo "2.0" ;
    dcat:landingPage
          <https://www.myorganization.org/website/ontologies> ;
    dcat:accessURL
          <https://sparql.myorganization.org/query> ;
    dcat:downloadURL
          <https://myorganization.org/ontologies/example_2.0.owl> ,
          <https://myorganization.org/ontologies/example_2.0.csv> .
```

4 Summary check-list when creating a new ontology version

In **Table 1**, we summarize the metadata related actions to do when creating a new ontology version. This can be used as a "check list". We assign a recommendation (M for Must and R for recommended) based on our appreciation of the importance.

5 Extension of the recommendations to any digital object

In this section (**Table 2**), we generalize our recommendations to extend them to any kind of digital objects. As done with MOD for semantic artefact, here we cannot rely on a harmonized metadata model that would work for any kind of digital objects; however, we can rely on multiple general metadata vocabularies such as Dublin Core, DCAT or Schema.org that will apply to multiple types of digital resources. Another challenge is to deal with objects not necessarily described or encoded with semantic web technologies – i.e., with no URIs.

Table 1. Check list of TODOs when creating a 'new' ontology version from a 'previous' version.

TODO in previous version	Rec.	TODO in new version	Metadata property
	M	□ Assign the same URI than the previous version.	Ontology IRI
	R	□ Duplicate the URI with an explicit metadata property.	`mod:URI`
	M	□ Assign a specific resolvable version URI.	`owl:versionIRI`
	R	□ Check the external PID provider policy if a new identifier is required	`dct:identifier`
	M	□ Update version info (following convention e.g., Semver)	`owl:versionInfo`
	M	□ Include a description, possibly independent of the versioning.	`dct:description`
	R	□ Include a comment or note specific to this version.	`rdfs:comment`
	R	□ Include a list of the changes or reference to a resource that describes the changes between previous and new version.	`vann:changes`
□ Update previous status in consequence if needed.	R|M	□ Change or maintain the status.	`mod:status`
□ Depreciate the previous version(s) (`true`) if necessary.	R|R	□ Assign depreciation flag (mostly `false`).	`owl:deprecated`
	M	□ Assign the same creation date than the previous version; except if major change requires a new time stamp.	`dct:created`
	M	□ Assign the current date as modification date.	`dct:modified`
□ Update the validity date to the creation date of the new version if necessary.	R|R	□ If you already know the future date of the next version, assign new version an end of validity date.	`dct:valid`
	M	□ Assign 'prior version'.	`owl:priorVersion`
	R	□ Inform of backward compatibility or incompatibility with previous versions.	`owl:backwardCompatibleWith` `owl:incompatibleWith`
	R	□ Add the previous version to the list of previous.	`dct:hasVersion`
	M	□ Include a landing page, possibly independent of the versioning.	`dcat:landingPage`
	R	□ Assign one or several access URL where the new version can be accessed / browsed / queried / visualized.	`dcat:accessURL`
	M	□ Assign one or several download URL where the new version can be downloaded in multiple formats.	`dcat:downloadURL`

The recommendations that hold for other digital objects are listed below including the metadata property used for semantic artefact (in bold before ":") and other possible properties to use (listed after ":"):

- **dct:description**: dct:description; schema:description
- **rdfs:comment**: rdfs:comment adms:versionNotes
- **dct:created**: dct:created, pav:createdOn, prov:generatedAtTime, schema:dateCreated
- **dct:modified** : dct:modified, pav:lastUpdateOn, schema:dateModified
- **dct:valid**: dct:valid, prov:invalidatedAtTime, schema:temporalCoverage
- **owl:priorVersion**: dcterms:isVersionOf, prov:wasRevisionOf, adms:prev
- **dct:hasVersion** : dct:hasVersion, pav:hasVersion
- **dcat:landingPage**: dcat:landingPage, foaf:page, vann:usageNote
- **dcat:accessURL**: :accessURLschema:url
- **dcat:downloadURL**: dcat:downloadURL, schema:distribution

Table 2. Extending the versioning recommendations (**Table 1**) to any digital objects using alternative metadata properties (not exhaustive).

Metadata property used for semantic artefact	**Comment / Generalization**	**Other possible property to use**
Ontology IRI, `mod:URI`, `owl:versionIRI`, `dct:identifier`	If the object does not have a URI, identification fully relies on the PID assigned by an external body. Some identifiers scheme does support versioning e.g., the HAL publication archive includes the version number at the end of the identifier[14] other data repository recommends specific policies e.g., Zenodo [21]. Thus, follow or check the PID provider policy if a new identifier is required for this new version of the digital object.	`dct:identifier` `schema:identifier`
`owl:versionInfo`	This recommendation holds for other digital objects.	`schema:version` `pav:version`
`vann:changes`	This recommendation holds for other digital objects; except that the type of changes and/or the formal language to express them will be different.	`vann:changes`
`mod:status`	These recommendations hold for other digital objects; except that the list of possible status might be different.	`adms:status` `idot:state`
`owl:deprecated`	These recommendations hold for other digital objects; the `owl:deprecated` property cand even be used as it can be applied to any web resource.	`owl:deprecated` `idot:obsolete`
`owl:backwardCompatibleWith`, `owl:incompatibleWith`	These recommendations are specific to ontologies or semantic artefacts.	`N/A`

[14] e.g., https://hal-lirmm.ccsd.cnrs.fr/lirmm-03208544v2 is the versioned identifier of a citation mainly identified with https://hal-lirmm.ccsd.cnrs.fr/lirmm-03208544

6 Conclusion

We have presented a set of recommendations to follow when versioning an ontology or a semantic artefact developed using Semantic Web technologies. We have illustrated pieces of information to properly encode with 20 metadata properties to semantically describe versioning. We have also shown the logics and dependencies between metadata properties. We suggest that the recommendations can be generalized to any digital objects that need to be versioned and semantically described, although further studies will be required for each type of objects. In AgroPortal, we are currently revisiting the metadata model to reinforce the recommendations discussed here typically by assigning unambiguous types and applying validators to property values. We are also continuing the metadata curation work with the editorial team and report to and/or try to involve the ontology developers.

Acknowledgement

This work has been supported by the *Data to Knowledge in Agronomy and Biodiversity* project (D2KAB – www.d2kab.org – ANR-18-CE23-0017), the Horizon Europe FAIR-IMPACT project (grant #101057344) and COGITO project (grant # 958310).

References

1. Dutta, B., Toulet, A., Emonet, V., Jonquet, C.: New Generation Metadata vocabulary for Ontology Description and Publication. In: Garoufallou, E., Virkus, S., and Alemu, G. (eds.) 11th Metadata and Semantics Research Conference, MTSR'17. , Tallinn, Estonia (2017). https://doi.org/10.1007/978-3-319-70863-8_17.
2. Jonquet, C., Toulet, A., Arnaud, E., Aubin, S., Dzalé Yeumo, E., Emonet, V., Graybeal, J., Laporte, M.-A., Musen, M.A., Pesce, V., Larmande, P.: AgroPortal: A vocabulary and ontology repository for agronomy. Comput Electron Agric. 144, 126–143 (2018). https://doi.org/10.1016/j.compag.2017.10.012.
3. Vandenbussche, P.-Y., Atemezing, G.A., Poveda-Villalon, M., Vatant, B.: Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web. Semant Web. 1, 1–5 (2014). https://doi.org/10.3233/SW-160213.
4. Jonquet, C., Toulet, A., Dutta, B., Emonet, V.: Harnessing the power of unified metadata in an ontology repository: the case of AgroPortal. Data Semantics. 7, 191–221 (2018). https://doi.org/10.1007/s13740-018-0091-5.
5. Amdouni, E., Bouazzouni, S., Jonquet, C.: O'FAIRe: Ontology FAIRness Evaluator in the AgroPortal Semantic Resource Repository. 19th Extended Semantic Web Conference, Poster and demonstration, Hersonissos, Greece. pp.89-94,. 13384 LNCS, 89–94 (2022). https://doi.org/10.1007/978-3-031-11609-4_17.
6. Amdouni, E., Bouazzouni, S., Jonquet, C., O'faire, C.J.: O'FAIRe makes you an offer: Metadata-based Automatic FAIRness Assessment for Ontologies and Semantic Resources. (2022). https://doi.org/10.13039/501100001665.
7. Poveda-Villalón, M., Espinoza-Arias, P., Garijo, D., Corcho, O.: Coming to Terms with FAIR Ontologies. In: Keet, M.C. and Dumontier, M. (eds.) 22nd International Conference on Knowledge Engineering and Knowledge Management, EKAW'20. pp. 255–270. Springer Science and Business Media Deutschland GmbH, Bolzano, Italy (2020). https://doi.org/10.1007/978-3-030-61244-3_18.

8. Garijo, D., Poveda-Villalón, M.: Best Practices for Implementing FAIR Vocabularies and Ontologies on the Web. In: Cota, G., Daquino, M., and Pozzato, G.L. (eds.) Applications and Practices in Ontology Design, Extraction, and Reasoning. IOS Press (2020). https://doi.org/10.3233/SSW200034.
9. Coxid, S.J.D., Gonzalez-Beltranid, A.N., Magagna, B., Marinescu, M.-C.: Ten simple rules for making a vocabulary FAIR. (2021). https://doi.org/10.1371/journal.pcbi.1009041.
10. Marc Novakouski, Grace Lewis, William Anderson, Jeff Davenport: Best Practices for Artifact Versioning in Service-Oriented Systems. (2022).
11. Klein, M., Fensel, D.: Ontology versioning on the Semantic Web. In: First International Conference on Semantic Web Working Symposium (SWWS'01). pp. 75–91. , Stanford (2001).
12. Noy, N.F., Musen, M.A.: Ontology versioning in an ontology management framework. IEEE Intell Syst. 19, 6–13 (2004). https://doi.org/10.1109/MIS.2004.33.
13. Michel Klein, Atanas Kiryakov, Damyan Ognyanoff, Dieter Fensel: Finding and specifying relations between ontology versions. In: Workshop on Ontologies and Semantic Interoperability. , Lyon, France (2022).
14. Auer, S., Herre, H.: A versioning and evolution framework for RDF knowledge bases. In: Virbitskaite, I. and Voronkov, A. (eds.) 6th International A. Ershov Memorial Conference, PSI'06, LNCS 4378. pp. 55–69. Springer, Novosibirsk, Russia (2007). https://doi.org/10.1007/978-3-540-70881-0_8.
15. Patel, A., Jain, S.: Ontology Versioning Framework for Representing Ontological Concept as Knowledge Unit. In: International Semantic Intelligence Conference (ISIC 2021). pp. 114–121. CEUR, New Delhi, India (2021).
16. Herbert Van de Sompel, Robert Sanderson, Michael Nelson, Lyudmila Balakireva, Harihar Shankar, Scott Ainsworth: An HTTP-Based Versioning Mechanism for Linked Data. In: C. Bizer, T. Heath, T. Berners-Lee, and M. Hausenblas (eds.) Workshop on Linked Data on the Web (LDOW2010). CEUR, Vol. 628, Raleigh, North Carolina, USA (2010).
17. Fiorelli, M., Pazienza, M.T., Stellato, A., Turbati, A.: Version control and change validation for RDF Datasets. In: 11th International Conference on Metadata and Semantic Research (MTSR'17). pp. 3–14. Springer, Tallinn, Estonia (2017). https://doi.org/10.1007/978-3-319-70863-8_1.
18. Garijo, D., Poveda-Villalon, M.: Best Practices for Implementing FAIR Vocabularies and Ontologies on the Web. 39–54 (2020). https://doi.org/10.3233/SSW200034.
19. Garijo, D.: WIDOCO: A wizard for documenting ontologies. In: 16th International Semantic Web Conference, ISWC'17. pp. 94–102. Springer, Vienna, Austria (2017). https://doi.org/10.1007/978-3-319-68204-4_9.
20. Suarez-Figueroa, Hartmann, J., Sure, Y., Haase, P., Suarez-Figueroa, M.: OMV–ontology metadata vocabulary. In: Welty, C. (ed.) Workshop on Ontology Patterns for the Semantic Web, WOP'05. p. 9. Springer, Galway, Irland (2005).
21. Nowak, K., Ioannidis, A., Bigarella, C., Nielsen, L.H.: DOI Versioning Done Right. (2018). https://doi.org/10.5281/ZENODO.1256592.