Glossary maturity levels

This page proposes maturity levels for *sharable glossary* use cases, along with expected fields for each level's schema profile.

Last updated: January 2022
Version: 0.1
Status: Unreviewed

This document is shared publicly.

Maturity levels

A glossary should start small, with minimal metadata fields. As the glossary matures, more metadata fields can be added to address advanced use cases.

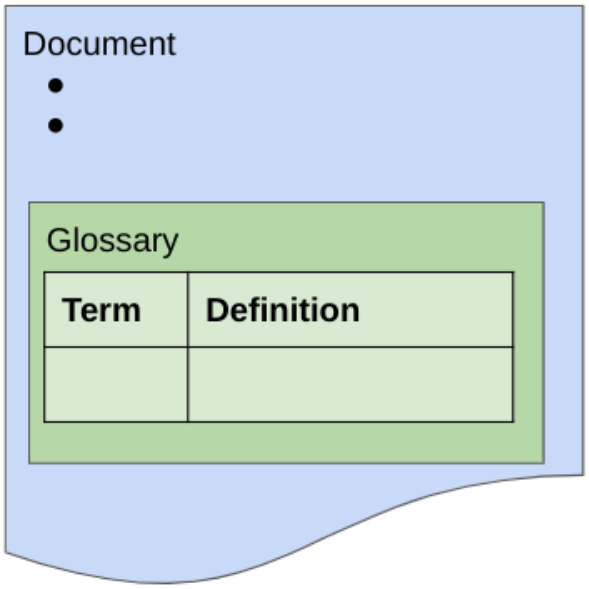
Glossary maturity levels

Level	Value added
0 No glossary	
1 Non-sharable	Help a reader understand locally defined terms.
2 Machine readable	Enables tooling, like domain specific hover-over popups.
3 Traced to source	Know which glossaries terms come from.
4 Governed	Manage glossary maintenance
5 Extra metadata	Support power use cases

Shared glossary maturity levels

Level 1: Non-sharable

What
A non-sharable glossary may be included as a table within a document or website.



Value added
A glossary is relatively straightforward to establish for a document or small website. It help readers understand terms used.

How
Copy domain-specific terms from your documentation into a glossary table. Include columns for Term and Definition. Ideally, source definitions from a more authoritative glossary.

Example

Term	Definition
mercury	metal element which is liquid standard temperatures

Level 2: Machine readable and reusable

What

- Store terms in a standard, machine readable format. Typically the glossary is stored as a static file, accessible via a URL, such as: https://mywebsite.org/glossary.json.



Value added
Adding machine readability enables multiple domain specific use cases, such as:

- Hover-over popups for terms within a document.
- Downstream glossaries referencing terms within this glossary.

How

- Level 2 of the *shared glossary profile* involves copying the terms and definitions from your Level 1 table into a glossary file. (Fields from higher maturity levels may additionally be included as needed.)

Fields for level 2

- skos:Collection # Glossary's top level, covering a collection of terms.
- skos:Concept # A term to describe
- skos:prefLabel@en # The preferred term name
- skos:prefLabel@fr # Optional: Other languages can be supported
- skos:altLabel # Optional: Store acronyms and synonyms
- skos:definition # Textual definition

Fields from higher maturity levels may optionally be included.

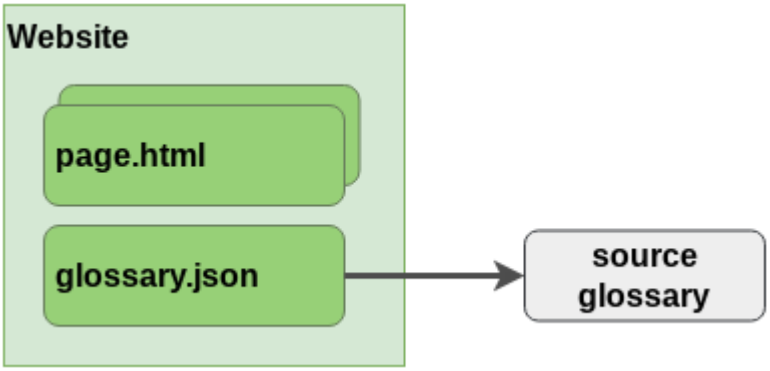
Example

```
{
  ex:chemistry rdf:type skos:Collection : {
    ex:mercury rdf:type skos:Concept : {
      skos:prefLabel "mercury"@en;
      skos:prefLabel "mercure"@fr; # Optionally support multiple languages
      skos:altLabel "he"@en; # altLabel is optional
      skos:definition "metal element which is liquid standard temperatures"@en;
  }
}
```

Level 3 Reference source glossaries

What

- Source terms from upstream glossaries and retain hyperlink to the source.
- Assign a license which allows repurposing of the glossary by downstream users, such as Creative Commons By Attribution (CC-By).



Value added

Adding a reference to the source glossary enables:

- Increased interoperability.
- Suitable attribution provided to address license conditions.
- Term definitions can be refreshed from updated source glossaries.
- A glossary's credibility will benefit from referencing a more authoritative source.

Extra fields for level 3

- dcterms:license
- skos:inScheme
- dcterm:identifier

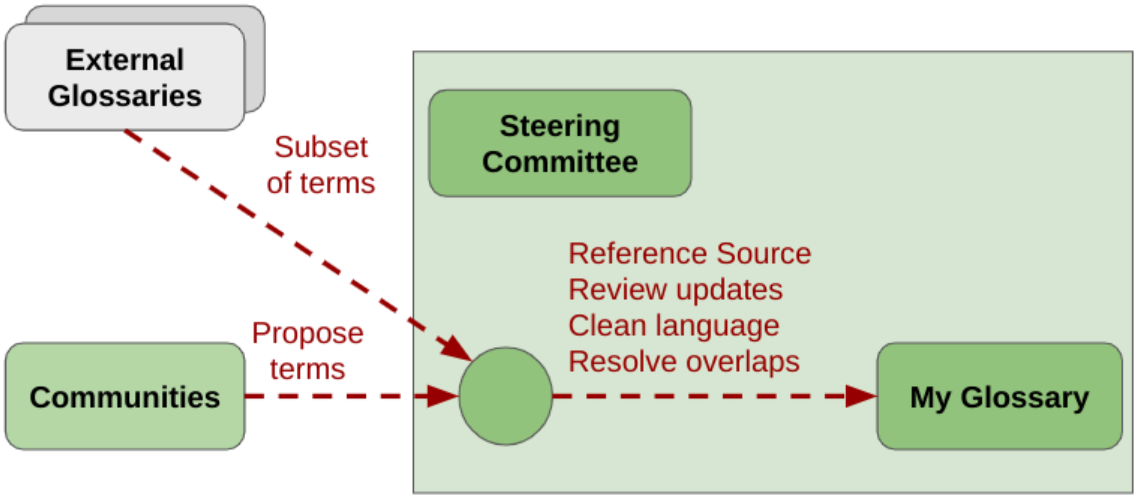
Example

```
{
  ex:chemistry rdf:type skos:Collection : {
    dcterms:license "https://creativecommons.org/licenses/by/4.0/";
    ex:mercury rdf:type skos:Concept : {
      skos:prefLabel "mercury"@en;
      skos:prefLabel "helio"@es; # Optionally support multiple languages
      skos:altLabel "he"@en;
      skos:definition "inert gas"@en;
      dcterm:identifier "mercury"; #unique identifier
      skos:inScheme "https://authoritative1.org/glossary.json"
      # If term doesn't have a source, then skos:inScheme isn't mentioned
  }
}
```

Level 4 Governed and versioned

What

- Apply governance processes to manage glossary updates.
- Identify a glossary custodian
- Apply a unique identifier for each term.
- Apply versioning to terms (skos:Concepts).
- Periodically release a baselined version of the glossary (skos:Collection).
- Publish glossary license information.



Value added

- Managed quality control processes.
- Planned release cycles.
- Reduced accidental inconsistencies.
- Maintainers can recommend updates to source glossaries.

Extra fields for level 4

- ex:collectionVersion
- ex:priorVersion
- ex:sourceGlossaryVersion
- ex:sourceConceptVersion

Example

```
{
  ex:chemistry rdf:type skos:Collection : {
    dcterms:license “https://creativecommons.org/licenses/by/4.0/”;
    ex:collectionVersion rdf:type owl:versionInfo “1.2.1”;
    ex: priorVersion rdf:type owl:priorVersion “1.2.0”;
    ex:mercury rdf:type skos:Concept : {
      skos:prefLabel "mercury"@en;
      skos:altLabel "he"@en;
      skos:definition "inert gas"@en;
      dcterm:identifier “mercury”; #unique identifier within Collection
      skos:inScheme “https://authoritative1.org/glossary.json”
      ex:sourceGlossaryVersion rdf:type owl:versionInfo “2.0”;
      ex:sourceConceptVersion rdf:type owl:versionInfo “3”;
}
```

Level 5 Extra metadata

More extensive use cases may involve adding extra metadata fields to the ConceptCollection and Concepts.
As at January 2022, this section requires further definition.

Possible extra fields for level 5

- skos:note # Comments about the term.
- skos:scopeNote # Clarifications on usage.
- dcterms:creator
- dcterms:created
- dcterms:modified
- dcterms:source
- dcterms:replaces
- rdfs:seeAlso
- skos:broader
- skos:narrower
- skos:related
- skos:broadMatch
- skos:closeMatch

- skos:exactMatch

- skos:narrowMatch

- skos:relatedMatch

- skos:ConceptScheme

Source:

Step	SKOS	OWL (basic)
Identify terms	Encode each vocabulary term as a skos:Concept, assigning an identifier as discussed in Rule 5	Encode each vocabulary term as an owl:Class, assigning an identifier as discussed in Rule 5
Encode term labels and synonyms	Encode the term name as the skos:prefLabel and synonyms and abbreviations in skos:altLabel. Language tags [20] can be used for multilingual vocabularies.	Encode the term name as rdfs:label. For synonyms you can use the SKOS elements skos:prefLabel and skos:altLabel. Note that some communities have their own terminology for labels and annotations (e.g. the OBO Foundry relies on the Information Artifact Ontology). Language tags [20] can be used for multilingual terms.
Add textual definitions	Encode the textual definition as skos:definition	Encode the textual definition as rdfs:comment
Add codes and symbols	Add any code or symbol as skos:notation (this applies if a formal code or symbol for the term is available, in addition to the name used for the skos:prefLabel)	Add any code or symbol as additional labels (rdfs:label)
Add notes or comments for clarifications	Comments can be encoded using skos:note. Clarifications on usage can be recorded using skos:scopeNote	Comments and clarifications can be encoded in the rdfs:comment
Add per-term metadata, if available	Individual terms may be annotated using standard elements such as dcterms:creator, dcterms:created, dcterms:identifier, dcterms: modified, dcterms:source, dcterms:replaces, rdfs:seeAlso. Per-term annotations are needed when they differ from values associated with the vocabulary as-a-whole (see Rule 7).	The same metadata elements can be used to annotate terms in the OWL encoding as well as owl:versionInfo, rdfs:comment, rdfs: isDefinedBy. Alternatively, adopt a specific solution for describing term metadata such as the OBO Metadata Ontology (http://www.obofoundry.org/ontology/omo.html)
Define the hierarchy of terms	If hierarchical relationships between terms are provided in the source document, encode these using skos:broader and skos:narrower. A narrower concept or subclass may be related to more than one broader concept or parent class, so each term may appear in more than one place in a hierarchy.	If hierarchical (is-a-kind-of) relationships between terms are provided in the source document, encode these using rdfs: subClassOf. A more specific concept or subclass may be related to more than one broader concept or parent class, so each term may appear in more than one place in a hierarchy. N.B. OWL sub-class relationships have more precise semantics than SKOS narrower/ broader relations.
Encode relationships between terms	If other relationships (non-hierarchical) between terms *within* the vocabulary are provided in the source document, they may be encoded using skos:related. Relationships to terms in *other* vocabularies (mappings) may be encoded using skos:broadMatch, skos:closeMatch, skos:exactMatch, skos:narrowMatch, skos:relatedMatch	dcterms:relation may be used to indicate related resources. However, it is usually better to use OWL object properties with the specific required semantics. It is recommended to re-use existing elements (e.g. relations ontology http://www.obofoundry.org/ontology/ro.html), or create your own if no existing one fulfils your requirement. owl:equivalentClass may be used for mappings.
Define subsets	If subsets or other groupings of terms are present in the source, encode each as a skos:Collection. Collections may be nested. Concepts may be members of more than one collection.	If subsets or other groupings of terms are present in the source, encode each as a class whose members are sub-classes
Define the whole vocabulary	The complete vocabulary should be encoded as a skos: ConceptScheme. Every skos:Concept should have a skos:inScheme relationship to the scheme, else the top terms in broader/narrower chains should have a skos:topConceptOf relationship to the concept scheme.	The complete vocabulary should be encoded as an owl:Ontology. Every member term that is not in the same namespace as the ontology should have a rdfs:isDefinedBy relationship to the ontology

https://doi.org/10.1371/journal.pcbi.1009041.t002

Image source in: Rules for making a vocabulary Findable Accessible Interoperable and Reusable

Further reading

- Rules for making a vocabulary Findable Accessible Interoperable and Reusable (FAIR)

Implementation challenges

Implementation support for metadata specifications

Many established metadata specifications, such as SKOS, Dublin Core Terms, DCAT, don't have implementation support for newer metadata specifications such as JSON-LD and SHACL.
Publishing authoritative "proxies" for these would reduce effort, inconsistencies and provide explicit interoperability information across similar usages.

Namespace clashes

TBD Rob: Can you please expand on what the current problem is with namespace clashes.

- What's wrong?
- What needs to be done to fix it?

Scalability of schema.org?

To help search optimization, search engine providers, such as Google, Microsoft and Yahoo, recommend including schema.org structured metadata in webpages.
Schema.org provides a centralized hierarchy of things and data types. A centralized metadata model is good for addressing core use cases, but becomes unwieldy as use cases scale. This is because:

- To cover many metadata use cases, schema.org will need to become large, resulting in large, unwieldy local caches..
- Many updates to schema.org may result in multiple version updates which would become hard to manage.

An alternative to centralization is to define metadata schemas as a distributed framework. A metadata profile (such as a schema for glossaries) is published at its own URL. It can reference other schemas as needed. This distributed pattern is similar to web pages within the internet.

Paywall restrictions for schema standards

Some standards, such as those provided by ISO, are legally restricted, held behind a paywall, which hinders adoption, especially to open-source communities which are often volunteer based.
Some fields which could be used within a glossary schema are potentially encumbered by legal paywalls.
This is an issue which will need to be considered. Possible options include:

- Requesting legal council to assess allowed options and fair use.
- Requesting standards owners to provide a legal waiver.
- Publishing unencumbered equivalent standards.

Further reading

- Sharable glossaries roadmap
- Glossary use cases
- Rules for making a vocabulary Findable Accessible Interoperable Reusable (FAIR)