The eNanoMapper Ontology

An application ontology to enable data integration for nanomaterial risk assessment

Javier Millán Acosta

BiGCaT - Maastricht University

September 19, 2022

Overview

1. Introduction
   - eNanoMapper
   - Engineered nanomaterials and safety
   - Ontologies
2. The eNanoMapper Ontology
   - Design of the eNanoMapper ontology
   - Uses of the eNM ontology
3. Future plans and challenges
   - What is still needed
   - Migrating the ontology development?

Overview

eNanoMapper

- **eNanoMapper** is a broader European project which aims to address data and model interoperability challenges for data management for engineered nanomaterial safety.
- **The eNanoMapper ontology** is an application ontology and reuses parts of several ontologies to describe the full domain of nanomaterial safety assessment.

Figure 1: The eNanoMapper logo

Engineered nanomaterials and safety

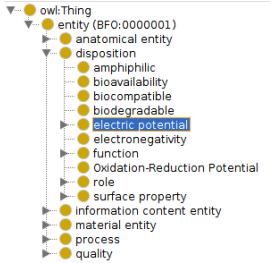
- Engineered nanomaterials (eNMs) are broadly defined as compounds that exist on a scale of 1–100 nm.in at least one of their dimensions.
  https://www.sciencedirect.com/topics/engineering/engineered-nanoparticles
- Their safety assessment must cover the identification of:
  - Physicochemical properties
  - Biological properties

  https://www.beilstein-journals.org/bjnano/content/pdf/2190-4286-6-165.pdf

Ontologies

- A formal representation of a set of concepts within a knowledge domain which can be used to a) define such domain; and b) reason about its properties **?**
- It consists of three syntactical categories: **Entities**, **Expressions** and **Axioms**, which can be given annotations for further description.

Figure 2: A class (highlighted in blue) in a partial hierarchy view of eNM.



Ontologies

- All entities (classes, object properties, named individuals...) are uniquely identified by a sequence of characters called IRI.

Figure 3: An owl:Class in an ontology text file (the ontology document). IRIs in blue.

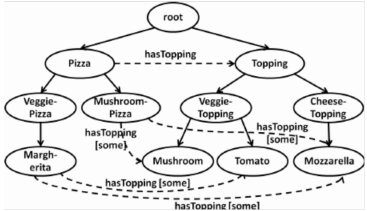
```
<!-- http://purl.obolibrary.org/obo/PATO_0001464 →

<owl:Class rdf:about="http://purl.obolibrary.org/obo/PATO_0001464">
<rdfs:subClassOf rdf:resource="http://purl.obolibrary.org/obo/PATO_0001018"/><rdfs:subClassOf
rdf:resource="http://purl.obolibrary.org/obo/BFO_0000016"/>
<obo:IAO_0000115 rdf:datatype="http://www.w3.org/2001/XMLSchema#string">A quality that is equal to the
potential energy per unit charge associated with a static (time-invariant) electric field, also called the electrostatic
potential.</obo:IAO_0000115>
<oboInOwl:hasOBONamespace rdf:datatype="http://www.w3.org/2001/XMLSchema#string
">quality</oboInOwl:hasOBONamespace>
<oboInOwl:id rdf:datatype="http://www.w3.org/2001/XMLSchema#string">PATO:0001464</oboInOwl:id>
<oboInOwl:inSubset rdf:resource="http://purl.obolibrary.org/obo/pato#attribute_slim"/><oboInOwl:inSubset
rdf:resource="http://purl.obolibrary.org/obo/pato#scalar_slim"/>
<rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">electric potential</rdfs:label>
</owl:Class>
```

Ontologies

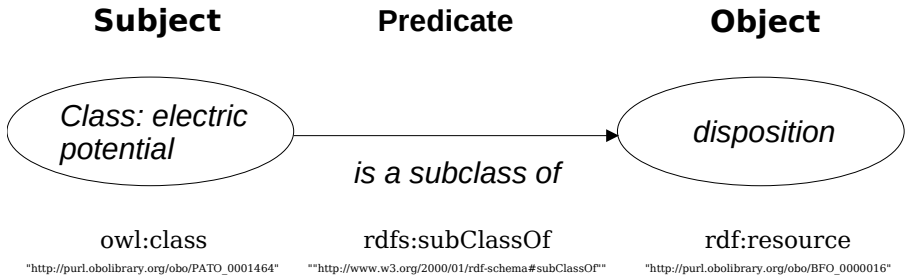
- Most ontologies use the W3C standard language for ontologies, Web Ontology Language **OWL**.
- OWL ontologies are mainly stored in `.owl` files, which are a sort of **RDF** document.
- **RDF** (Resource Description Framework) is a standard for data exchange. It defines **triples** of **(subject, predicate, object)**.
- These triples form labeled graphs where the edge (predicate) represents the link between two resources (subject and object)

Figure 4: The pizza ontology, visualized as a graph (add source)



Ontologies

Figure 5: A graph with two nodes (Subject and Object) and a triple connecting them (Predicate)



Ontologies

Figure 6: The triple in the previous figure as expressed in the `.owl` document file of the ontology it is contained in.

```
<!-- http://purl.obolibrary.org/obo/PATO_0001464 -->

<owl:Class rdf:about="http://purl.obolibrary.org/obo/PATO_0001464">
<rdfs:subClassOf rdf:resource="http://purl.obolibrary.org/obo/BFO_0000016"/>

 (...)
</owl:Class>
```

Ontologies

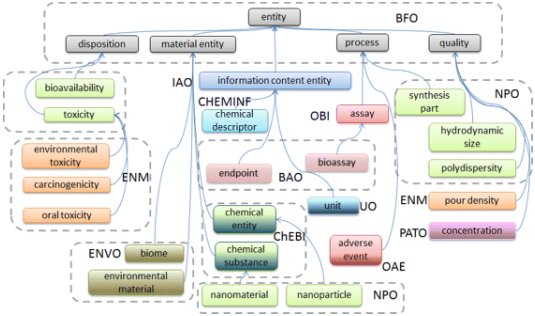
- **Foundation ontologies**: they provide the most abstract or general classes, i.e., the top-level classes we see in a hierarchy view of our ontology.
- **Application ontologies**:
- **Domain ontologies**

Overview

1. Introduction
2. The eNanoMapper Ontology
   - Design of the eNanoMapper ontology
   - Uses of the eNM ontology
3. Future plans and challenges

Design of the eNanoMapper ontology

Figure 7: AN overview of the upper levels, imports and manually annotated content going into the eNanoMapper ontology



https://faircookbook.elixir-europe.org/content/recipes/interoperability/ontology-robot-recipe.html make a similar figure!

Uses of the eNM ontology

Pointers about this...how do we add terms, etc

Overview

What is still needed

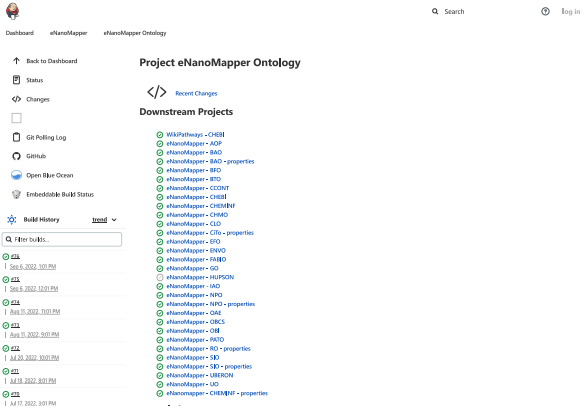
- OWL is not the best at modularity, but the eNM ontology heavily relies on importing modules. This leads to complications in class hierarchies, duplicate imports, etc.
- This peculiarity

Migrating the ontology development?

- Current setup: Jenkins for CI
- Alternatives:
  - Literal migration from Jenkins to GitHub actions (ref figure)
  - The Ontology Development Kit (ref figure)

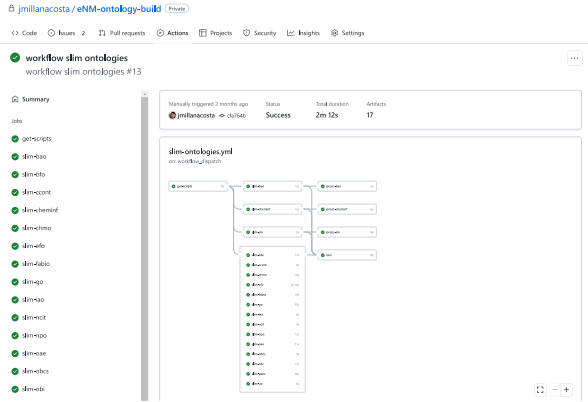
Migrating the ontology development?

Figure 8: The current Jenkins setup for eNanoMapper ontology CI



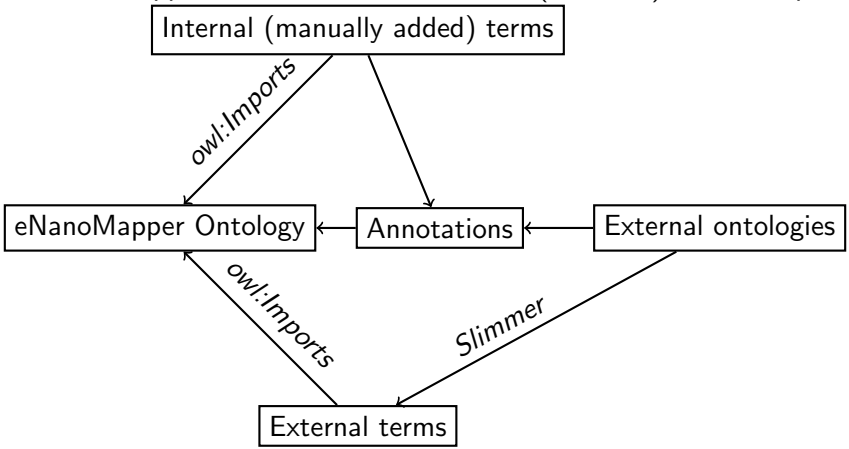
Migrating the ontology development?

Figure 9: A test run for the GitHub actions-based setup for eNanoMapper ontology CI



Migrating the ontology development?

The current approach uses in-house software (*Slimmer*) to slim imports.



Migrating the ontology development?

ROBOT, an OBO-centered software, could replace *Slimmer*

Migrating the ontology development?

ODK diagram figure?

References I