DELIVERABLE REPORT D2.1

Framework and Infrastructure for ontology development, versioning and dissemination

GRANT AGREEMENT:	604134
ACRONYM:	eNanoMapper
NAME:	eNanoMapper - A Database and Ontology Framework for Nanomaterials Design and Safety Assessment
PROJECT COORDINATOR:	Douglas Connect GmbH
START DATE OF PROJECT; DURATION:	1 February 2014; 36 months
PARTNER(s) RESPONSIBLE FOR THIS DELIVERABLE:	UM, EMBL-EBI
DATE:	1.6.2014
VERSION:	V.1.0.

Call identifier	FP7-NMP-2013-SMALL-7
Document Type	Deliverable Report
WP/Task	2 - 1
Document ID	eNanoMapper D2.1.
Status	Final

Partner Organisations	• Douglas Connect, GmbH (DC) • National Technical University of Athens (NTUA) • In Silico Toxicology (IST) • Ideaconsult Ltd (IDEA) • Karolinska Institutet (KI) • VTT Technical Research Centre of Finland (VTT) • European Bioinformatics Institute (EMBL-EBI) • University of Maastricht (UM)
Authors	Janna Hastings (EMBL-EBI), Egon Willighagen (UM) Reviewed by Barry Hardy (DC)
Purpose of the Document	To report on the technical infrastructure that has been implemented for the eNanoMapper ontology development, dissemination and versioning.
Document History	1. First draft, 01/06/2014 2. Internal draft for review 15/06/2014 3. EW review and comments 30/06/2014 4. JH pre-final version 02/07/2014 5. Completed version, 21/07/2014 6. Final reviewed version 1.0, 31/07/2014

TABLE OF CONTENTS

TABLE OF FIGURES

GLOSSARY

Abbreviation / acronym	Description
WP	Work Package
OWL	Web Ontology Language
MIREOT	Minimum Information to Reference an External Ontology Term
API	Application Programming Interface
URL	Universal Resource Locator
BFO	Basic Formal Ontology
CHEMINF	Chemical Information Ontology
DC	Dublin Core Metadata Ontology
IAO	Information Artifact Ontology
NPO	NanoParticle Ontology
RO	Relations Ontology
BAO	BioAssay Ontology
OBI	Ontology of Biomedical Investigations

1. EXECUTIVE SUMMARY

The eNanoMapper project aims to build an ontology and database to collate and describe data relevant for “safe by design” engineered nanomaterial development. Work Package 2 of this effort will develop and disseminate a comprehensive ontology for the nanosafety domain, encompassing nanomaterials and all information relating to their characterization, as well as relevant experimental paradigms, biological interactions and safety information. This deliverable report describes the technical infrastructure that has been created in order to support the development, versioning and dissemination of the ontology. The technical infrastructure also supports the process of the re-use of existing ontologies, as the eNanoMapper objective is to fully harness pre-existing efforts thus avoiding unnecessary and divisive re-implementation. At the same time, re-use poses challenges, thus, our technical infrastructure has been created such as to support modular incorporation of external content from multiple sources while removing duplication and contradictions that can arise from integration. We have implemented an infrastructure for extensive unit and integration testing to ensure that changes in underlying source ontologies do not break the coherence of our offering. All of these aspects are described in this report.

2. INTRODUCTION

Nanomaterials are materials in which the units have at least one dimension sized in the 1-100nm range. In addition to the wide diversity of natural nanomaterials available, advances in chemical synthesis techniques have led to an explosion in engineered nanomaterials (ENMs) in recent years. Materials with structures in the nanoscale range often have unique optical, electronic, and mechanical properties, and as a result ENMs are being developed to meet specific application needs in diverse domains across the engineering and biomedical sciences (e.g. drug delivery). However, accompanying the proliferation of nanomaterials is a challenging race to understand and predict their possibly detrimental effects on human health and the environment.

The eNanoMapper project (www.enanomapper.net) is creating a pan-European computational infrastructure for toxicological data management for ENMs, based on semantic web standards and ontologies. eNanoMapper aims to develop a comprehensive ontology and annotated database for the nanosafety domain to address the challenge of supporting the unified annotation of nanomaterials and their relevant biological properties, experimental model systems (e.g. cell lines), conditions, protocols, and data about their environmental impact. Rather than starting afresh, the developing ontology will build on existing work, integrating existing ontologies in a flexible pipeline. The establishment of a universal standardisation schema and infrastructure for nanomaterials safety assessment is a key project goal, which will catalyze collaboration, integrated analysis, and discoveries from data organised within a knowledge-based framework. This framework will support the discovery of nanomaterial properties responsible for toxicity, and the identification of toxicity pathways and nano-bio interactions from linked datasets, ontologies, omics data and external data sources.

Ontologies are structured controlled vocabularies enhanced with explicit formal relationships between entities in support of advanced automated reasoning for inference and error detection. **Work Package 2** of the eNanoMapper project focuses on the development and dissemination of a comprehensive ontology for the nanosafety domain, encompassing nanomaterials and all information relating to their characterization, as well as relevant experimental paradigms, biological interactions and safety information.

As technical artifacts similar to items of software, ontologies require technical infrastructure to create, maintain and disseminate. Thus, the initial objective of Work Package 2 is to setup the technical framework within which the ontology can be developed. This deliverable report accordingly describes the technical infrastructure that has been created in order to support the development, versioning and dissemination of the ontology. The technical infrastructure also supports re-use of existing ontologies, as the eNanoMapper objective is to fully harness pre-existing efforts thus avoiding unnecessary and divisive re-implementation. At the same time, re-use poses challenges, thus, our technical infrastructure has been created such as to support modular incorporation of external content from multiple sources while removing duplication and contradictions that can arise from integration. We have implemented an infrastructure for extensive unit and integration testing to ensure that changes in underlying source ontologies do not break the coherence of our offering. All of these aspects are described in this report.

3. TECHNICAL INFRASTRUCTURE

3.1 STORAGE, DISSEMINATION AND VERSIONING

The primary hub within which the ontology will be stored and which enables both dissemination and versioning is provided for us by GitHub (http://www.github.com/). GitHub is a web-based version control and hosting system for collaboratively developed software or other technical projects. It provides file hosting within a project and nested directory structure, and files are versioned using the Git distributed versioning system architecture. Within GitHub, developers create logins and can be assigned rights on different repositories. Repositories may be forked and worked on independently, while changes can then subsequently be merged in to downstream repositories using the social push/pull infrastructure with comments and patch support. This supports the modular development, review, and maintenance of the ontology. The default git and GitHub application programming interfaces (APIs) allow integration into our development workflow.

An eNanoMapper project has been created on GitHub, which includes several repositories. This can be found at http://github.com/enanomapper/. Several repositories are listed on the project page, relating to all aspects of the eNanoMapper work for which a version control system is needed. For the purposes of ontology development, the important repositories are 'ontologies', in which the ontology is stored and versioned, and 'ontoTest', in which the continuous integration unit tests are implemented and further tests will be added in the future.

The GitHub infrastructure provides not only storage and versioning, but also one important avenue for dissemination of the developed ontology. The repositories and all of their content are publically available and may be downloaded in full by any interested user. While it might be objected that the programmer-oriented user interface of the GitHub system might deter users who are not from a technical background, we observe that many scientists within increasingly data-driven scientific paradigms including that of nanoscience are comfortable in technical environments. Moreover, the GitHub webpages provide a web interface for users to suggest changes and make comments, without leaving the browser. Finally, we stress that this is only one of several dissemination routes via which we will make the ontology available, and other routes will be tailored to different classes of user based on the outcomes of our user requirements analysis investigations (WP1).

3.1.1 ONTOLOGIES REPOSITORY

The 'ontologies' repository is structured as follows (See Figure 1). Within the main folder, there is a README.md file which describes the repository (as recommended by GitHub). There is also the main enanomapper.owl ontology file (which will be further described below). There are three directories organizing further content in subsidiary nested folders: 'external', which contains the content modules which have been extracted from external ontologies (as described below), 'internal', which contains internally developed content modules, and 'scripts', which is where the software modules needed for supporting the ontology development process are stored.

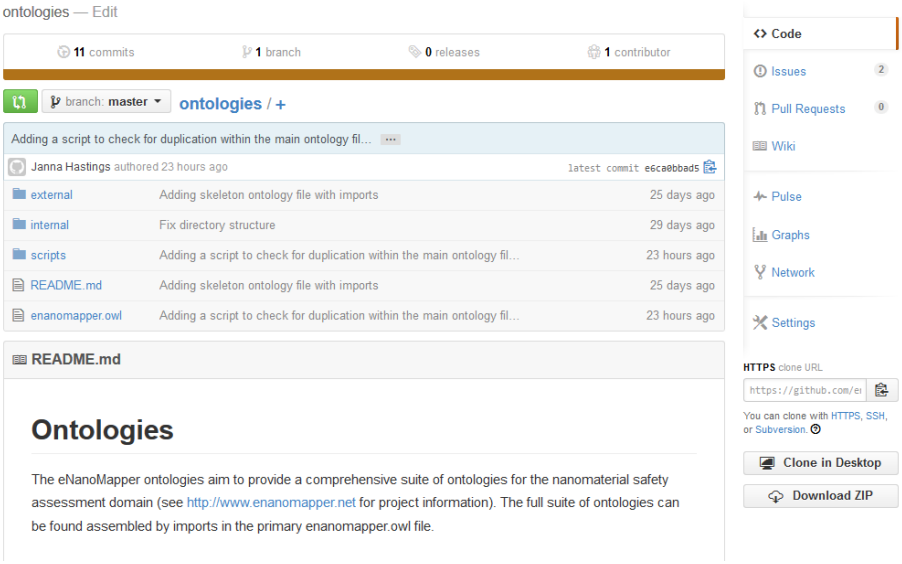

Figure 1: The eNanoMapper ontologies repository

Files within the repository can be accessed both via an online browser interface, which allows viewing of the commit history and any comments or open issues that have been logged, or as "raw" files which is the avenue through which the latest version of the content can always be accessed online. For example, an extract of the commit history for the enanomapper.owl ontology file is shown in Figure 2.

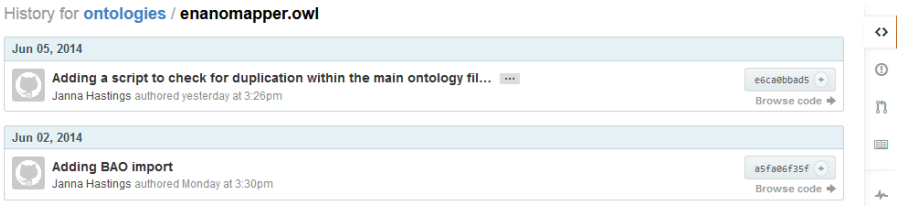

Figure 2: The eNanoMapper ontologies repository commit history

The URL to access the raw content of the enanomapper.owl file is https://raw.githubusercontent.com/enanomapper/ontologies/master/enanomapper.owl. This URL contains the path to the project and repository as well as the fork label ("master") of the file.

3.1.2 ONTOTEST REPOSITORY

The 'ontoTest' repository is structured as a general Java application using the Maven layout, including a src/test/java/ directory structure to contain the test classes. The test classes use the JUnit testing framework (http://junit.org/) to formalize requirements and expected behavior as isolated tests, i.e., each imported ontology can be separately tested, while integration tests can be run too. The currently tested ontologies are the Basic Formal Ontology (BFO, an upper-level organizing ontology); the Chemical Information Ontology (CHEMINF, for chemical descriptors and attributes); Dublin Core (DC, a small metadata ontology); the Information Artifact Ontology (IAO, for ontology metadata and categories of information entity); the NanoParticle Ontology (NPO); the Relationship Ontology of shared relations (RO); and our integrated eNanoMapper ontology. For each a separate JUnit test class is found in the https://github.com/enanomapper/ontoTest/tree/master/src/test/java/net/enanomapper/onto/test folder. Each test class extends a generic *AbstractOntologyTest* class that defines the tests to be run on

all ontologies. This class tests whether the OWL file is valid RDF/XML, can be parsed as an OWL ontology, and runs some general perception of common errors as defined by the OWLAPI (http://owlapi.sourceforge.net/). More tests will be added later using this same infrastructure. The test classes also define the mappings of ontology URIs to local instances of the OWL files, overcoming the problem that not all ontology OWL files can be automatically retrieved from the web using the respective URIs.

Implementation of the full testing platform is automated using the Jenkins (http://jenkins-ci.org/) build system software and is available at https://jenkins.bigcat.unimaas.nl/ (Fig. 3):

Figure 3: The Jenkins build system, eNanoMapper installation

For each ontology used by eNanoMapper, the testing platform defines a Jenkins job and in the above screenshot we see jobs for the various ontologies. Each “ontology” job downloads a recent copy of the ontology, and may do further integration later on, like defining slices of that ontology, if the eNanoMapper decided to incorporate only part of the ontology. The Ontology Testing jobs runs the unit tests on the various ontologies, as defined in the ontoTest repository on GitHub.

Jenkins allows jobs to be automatically run when changes are made to the code repositories in which the ontologies are stored. It is also configured to rerun the Ontology Testing job when such an event happens. The test report is provided on the Jenkins page (see Fig. 4), and provide detailed reports on why a particular unit test failed (not shown). eNanoMapper developers are notified when the number of failing unit tests changes.

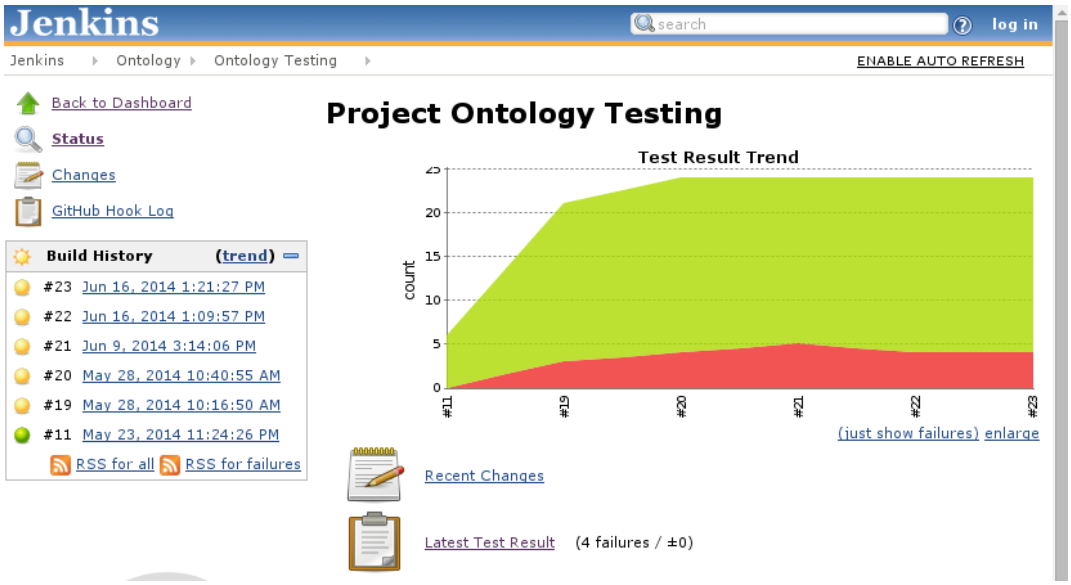

Figure 4: The Jenkins test results for the eNanoMapper ontology tests

3.2 ONTOLOGY DEVELOPMENT

3.2.1 DEVELOPMENT LANGUAGE

The ontology is being developed in the Web Ontology Language (OWL: W3C, 2012). OWL is a standard approved by the W3C for the representation of domain knowledge embedded in a semantic web framework. It is increasingly being adopted by the life sciences community for their ontology development efforts, and the provision of multiple ontologies in the same underlying language aids interoperability and convergence to a common framework.

OWL ontologies consist of classes, which represent entities in the modelled domain, and properties, which represent relationships. An example of a class is 'nanoparticle' while an example of a property is 'has_part'. Classes are arranged in a hierarchy using the built-in organizing relationship 'subClassOf'. If class A subClassOf class B, then A inherits all of B's properties. For example, 'quantum dot' is a subclass of 'nanosphere': it inherits all the properties of 'nanosphere' and has additional properties (in the size range). OWL is a powerful language in that it allows for logical statements to be built up from basic constructs (such as the logical operators 'and,' 'or' and 'not' and quantifiers such as 'only' and 'some') to capture a sophisticated model of the knowledge in the domain. OWL reasoners are software implementations that take an ontology expressed in OWL and compute inferences based on the captured knowledge. Inferences might enhance the knowledge captured, such as discovering additional subclass relationships that have not been explicitly expressed but can be deduced from the captured knowledge, or they may detect errors and inconsistencies, thus aiding the ontology development process.

A feature of the OWL language is that the ontology itself, and each item included in it, are embedded in a Web context through being identified with a URI which should map to a resolvable URL. For the eNanoMapper project, the ontology URIs are created in the URL space http://purl.enanomapper.org/onto/. Each class in the ontology is associated with a numeric identifier suffixed to the core class URI. For example, a class with id 23 is given the URI http://purl.enanomapper.org/onto/ENM_000023.

3.2.2 METADATA STANDARDS

The OWL language allows for the sophisticated capture of metadata in the form of annotations to all parts of the ontology. For eNanoMapper, of particular importance is the annotation of metadata at the class level in the ontology: names, synonyms, definitions, cross-references and curation status. The annotation properties that we use are supplied by the Dublin Core ontology (DC) supplemented with a subset of the Information Artifact Ontology (IAO) ontology-metadata ontology. Each class created in the eNanoMapper ontology is expected to have the following core metadata associated:

- A unique, unambiguous name, captured using the rdfs:label annotation property. The name should be unique within the ontology and should contain sufficient detail for a novice consumer of the ontology to understand the entity described at least in broad outline (i.e. no cryptic codes!)
- A text definition captured using the IAO's definition annotation property. The text definition should be one or more full sentences that fully describe the entity, with particular attention to individuation criteria, i.e. how does that entity differ from its immediate subsumption parent and its siblings on the same level? The definition should be complete enough to allow ontology users to determine precisely which class to use for annotation of their data. Optionally, the definition may be supported with examples of usage (another IAO metadata property) e.g. for specific descriptions of use in data annotation.
- Synonyms should be captured using the IAO 'alternative term' annotation property.
- Cross-references to databases may be captured using the IAO hasDbXref annotation property. Note: this method of cross-referencing is not to be used for other OWL ontologies in which the entity appears, as that renders the cross-reference invisible to the machinery of the OWL language and reasoners. Rather, for inter-ontology references bridging modules should be created in which the two classes are related using an appropriate OWL language construct, e.g. EquivalentClasses if they represent the exact same entity.
- To capture the curation status of a class and additional metadata related to the curation of the ontology, the IAO annotation properties 'definition editor', 'definition source' and 'has curation status' should be used. By default, if no definition editor, source and curation status are specified, it can be assumed that the eNanoMapper project WP2 has performed the curation and that the status is 'ready for public review'.

3.2.3 CURATION PLATFORM

Curation of the ontology is being performed using the Protégé ontology editor, illustrated in Figure 5 (http://protege.stanford.edu/). Protégé is the de facto standard free and open source OWL ontology editor. It allows visualization and editing of all aspects of an OWL ontology, and offers a plug-in framework for flexible extension within which many additional utilities are available. We use Protégé for editing classes, properties, annotations and axioms. It is also possible to run ontology reasoners within the Protégé framework and to view errors or save inferences back into the ontology.

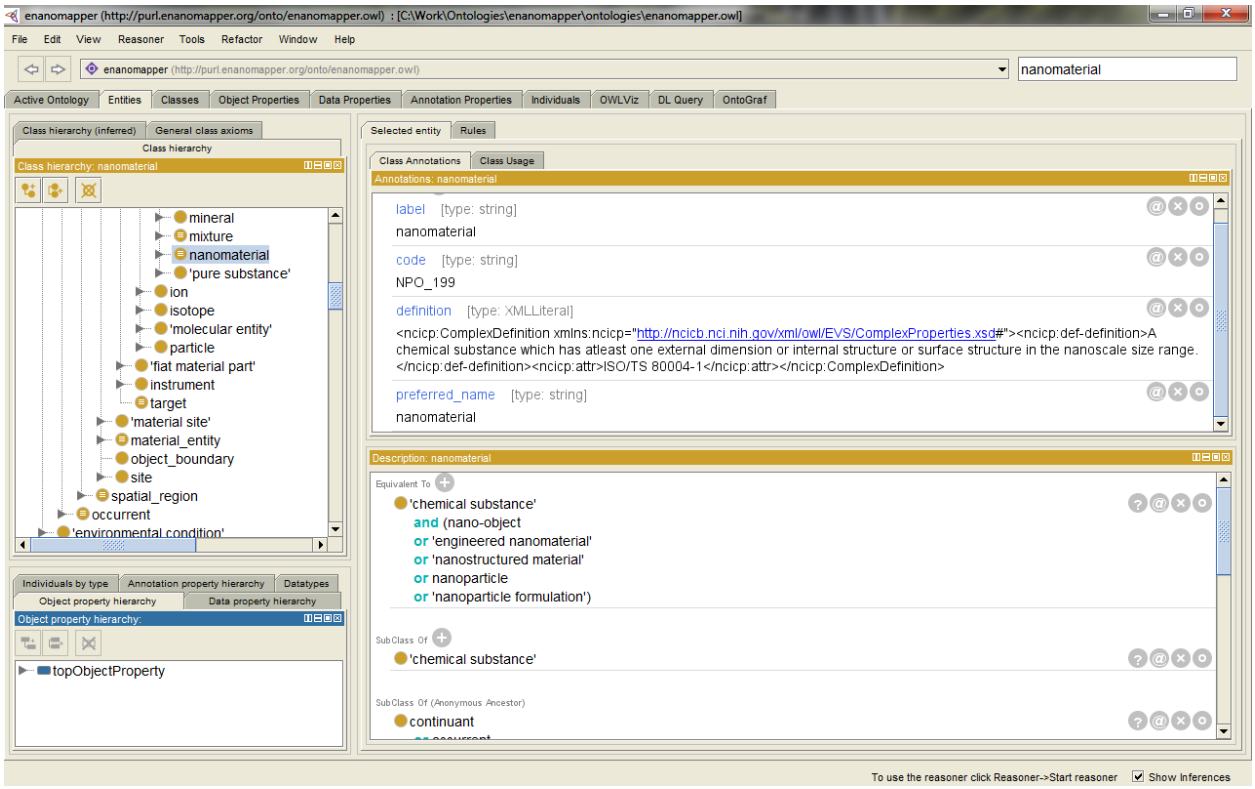

Figure 5: Ontology editing in Protégé.

3.3 RE-USING EXTERNAL CONTENT

The eNanoMapper project is committed to a policy of ontology re-use rather than re-invention. This means that wherever a reasonable candidate ontology can be found covering a relevant portion of domain content, that ontology or a relevant subset thereof should be included in the eNanoMapper fully assembled ontology using the OWL imports mechanism, and used in all relevant data annotations arising from the project. A full description of the target ontologies which are being re-used is out of scope for this technically-oriented deliverable report, but will be the subject of Deliverable D2.2.

External content may be imported into the ontology in two ways: simple or complex. In the simple case, the full external ontology is imported into the primary ontology file, as illustrated in Figure 6.

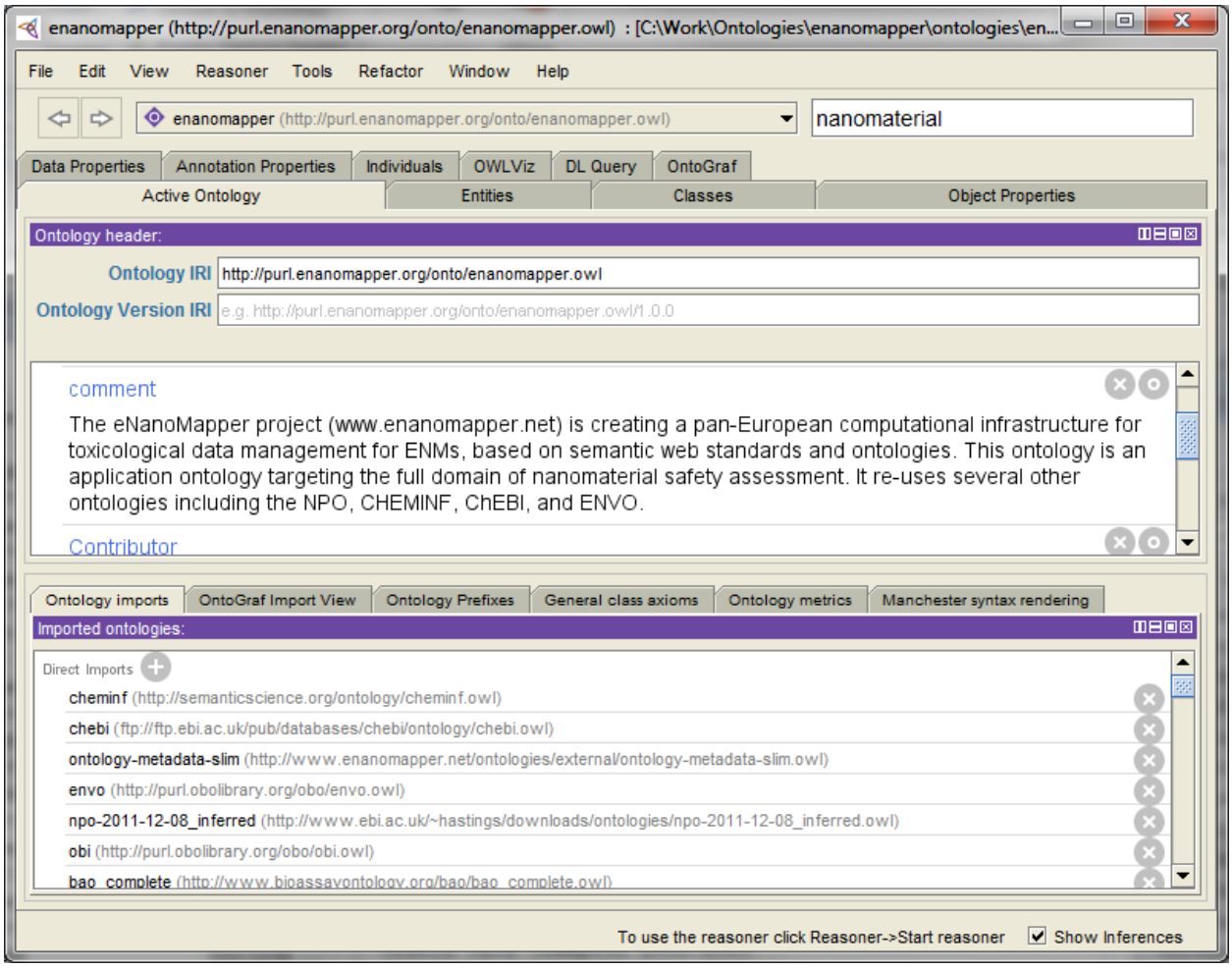

Figure 6: Ontology imports within Protégé

However, there are various reasons this may not be possible or preferred, including:

- The external ontology may include content which is irrelevant or incorrect which we would not like to include.
- The external ontology may include content which overlaps with the content included in another external ontology that we would also like to import (it is very important not to end up with duplicates within the final ontology!)
- The external ontology may be structured in a fashion that we do not fully agree with, such that we want to re-use their classes but not their hierarchy or relations.

In such cases, we will aim to follow the MIREOT methodology (Courtot *et al.*, 2011) for re-using external ontology content without fully importing the external ontology. Specifically, we will create a custom subset of the external ontology for our use, referenced extensively back to the original source ontology, and then import that subset. The subset creation can be done manually or automatically. Eventually we would need to automate the process fully in order to cope with updates of source ontologies and releases of our ontology; at the beginning of the project while we are evolving our scripts we may find that we must initially prepare files manually. The OntoFox tool (Xiang *et al.*, 2010) is of use in this process. OntoFox allows extraction of subsets from ontologies according to different parameters. For example, a root term can be specified and all descendants can be extracted into a module; parameters control whether to include relationships and annotations or not. The most important aspect of ontology-reuse in this fashion is that the URI of each class that is re-used is used as it is identically in the source ontology, and that the use is appropriately cited. For our purposes, it is important that the whole

process can be regularly updated, as we expect the source ontologies will change and we need to be able to seamlessly remain up-to-date.

Modules which are prepared for import should be appropriately named and identified (e.g. http://purl.enanomapper.org/onto/external/NCBITaxon_import.owl) and placed in the external folder in the GitHub repository. From the new location, the module can then be imported into the main ontology.

4. CONCLUSION

In this report we have described the technical infrastructure that has been put in place in order to facilitate the ontology development – Work Package 2 of the eNanoMapper project. While the infrastructure that we have created thus far is expected to remain stable for the duration of the project, we may find that as the project develops we add further technical infrastructure, including additional utility scripts and automated tests, additional software for the production pipeline and added plugins for facilitation of reporting. Therefore, this report may need to be updated during the lifecycle of the project. Furthermore, we undertake to ensure that all of our technical infrastructure is always appropriately and fully documented in the appropriate technical channels for documentation, e.g. code is appropriately commented, repositories have full and useful readme files, and so on.

5. BIBLIOGRAPHY

Courtot, M., Gibson, F., Lister, A. L., Malone, J., Schober, D., Brinkman, R. R., Ruttenberg, A. (2011) MIREOT: The minimum information to reference an external ontology term. *Journal of Applied Ontology*, Volume 6 Issue 1, Pages 23-33.

W3C OWL Working Group (2012), OWL 2 Web Ontology Language: Document Overview (Second Edition), *W3C Recommendation*, available at http://www.w3.org/TR/owl2-overview/, 11 December 2012.

Xiang, Z., Courtot, M., Brinkman, R. R., Ruttenberg, A., He, Y. (2010) OntoFox: web-based support for ontology reuse. *BMC Research Notes,* 3:175.