eagle-i ontology

The eagle-i Ontology

A unique feature of eagle-i is that the data collection and search tools are completely driven by ontologies. These ontologies are a set of modules that are written in the OWL language and edited and managed using Protégé. The following instructions will help you configure Protégé, edit the core ontology module, and add annotations to drive the user interfaces.

Goals

Our modeling approach had three main drivers. The first was to represent real data collected about resources. The second was to have the ontology control the userinterface (UI) and the logic of the data collection tool and search application. The third was a commitment to build a set of ontologies that could be reusable and interoperable with other ontologies and existing efforts for representing biomedical entities. This latter requirement translated into decisions to a) follow OBO Foundry principles and best practices for biomedical ontology development and b) engage in active discussions within the bio-ontology community in order to provide context for eagle-i interoperability and align with domain-wide standards for resource representation (http://bit.ly/rrcoord).

Ontology structure

Viewing the ontology using Protégé

The following instructions will help you configure Protégé, edit the core ontology module, and add annotations to drive the user interfaces. Protégé is a free, open source ontology editor and knowledge-base framework.

Note: You can *view* the Ontology using Protégé 4.0.2 but to edit, you must install Protégé 4.1.

Step 1: Download Protégé

1. Download OBI-friendly Protégé 4 for:
   Windows: http://dl.dropbox.com/u/4905123/Protege_Windows.zip
   Mac:http://dl.dropbox.com/u/4905123/Protege41MAC.zip
2. If prompted, do not install updates for the plugins, as the plugin versions contained in the download work specifically with OBI.

Step 2: Configure Protégé

1. Start Protégé and open an ontology. Any ontology will work.
2. A dialog may appear asking you about importing OWL files; if so, click Cancel.

Step 3: Set Protégé preferences

1. Open the File menu and click Preferences...
2. Click Renderer, and then click Annotations ..., add `'en, en-US, !'` in the languages column. Include the apostrophe.
   Note: OBI users may want to change the "reasoner" default settings to improve reasoning performance. In the reasoner tab "initialization" , choose only "class members" and under "Displayed inferences" choose only "unsatisfiable classes", "equivalent classes" and "superclasses". No instance reasoning is done, but for most OBI development this isn't necessary. These settings can make a particularly big difference when using Hermit as the reasoner.
3. Open the Save tab to confirm that 'Use XML Entities' is checked.
4. To display the class labels rather than the IDs, open Renderer and choose Render entities using annotation values. If the eagle-i classes still display as IDs, open the Renderer and click Annotations ..., then select label. Leave the language column blank and click OK.
5. Click OK.

Step 4: Add the local files for ontologies you want to import.

This step adds a set of directories from your local (Subversion) SVN repositories. This example uses the `/datamodel/` directory.

1. To begin, locate the SVN local copy on your machine.
2. Select File > Ontology Libraries from the menu.
   This step add directories. For each directory, click + and then navigate to the specific directory:

```
/datamodel/trunk/src/ontology/imports
/datamodel/trunk/src/ontology/external
/datamodel/trunk/src/ontology/external/iao
/datamodel/trunk/src/ontology/external/iao/external
/datamodel/trunk/src/ontology/external/iao/protégé
```

Your dialog window should look like this:

https://sites.google.com/a/eagle-i.org/workspace/data-curators-workspace/configuring-and-editing-with-protege-4/onto_lib.tiff?attredirects=0

1. Close Protégé.

Adding a New Term to the eagle-i Ontology

It is important to note that the following steps assume that you have a basic understanding of Protégé, and the revision control software Subversion as well as an understanding of an ontology.

There are seven basic steps when adding a new term to the ontology.

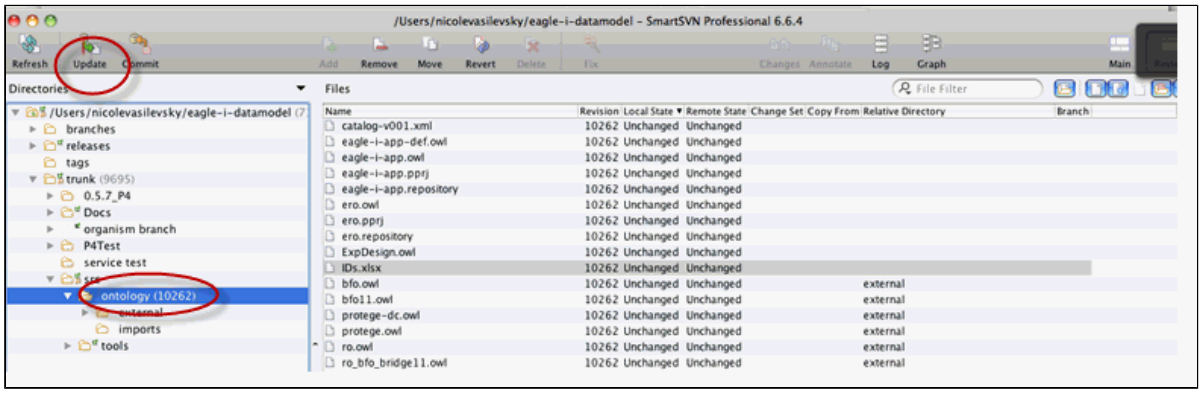
Step 1: Send an e-mail to notify the curators you are locking files.

1. You will be working with several files under Subversion (SVN) revision control. Before you begin, notify the curators by e-mail that you are locking two of these files: `ero.owl` and `eagle-i-app.owl`

The body of the e-mail should looks something like this:`[LOCKING] eagle-i-app.owl and ero.owl for editing.`

Step 2: Open Subversion and update the ontology files.

1. Open SmartSVN. The files you will be working with are located here: `svn+ssh://orchestra.med.harvard.edu/svn/cbmi/u24/eagle-i-dev/datamodel/trunk/src/ontology`
2. Click Update to update the entire trunk.



This step updates three files, `ids.xlsx, ero.owl` and `eagle-i-app.owl`, as well as all relevant imports.

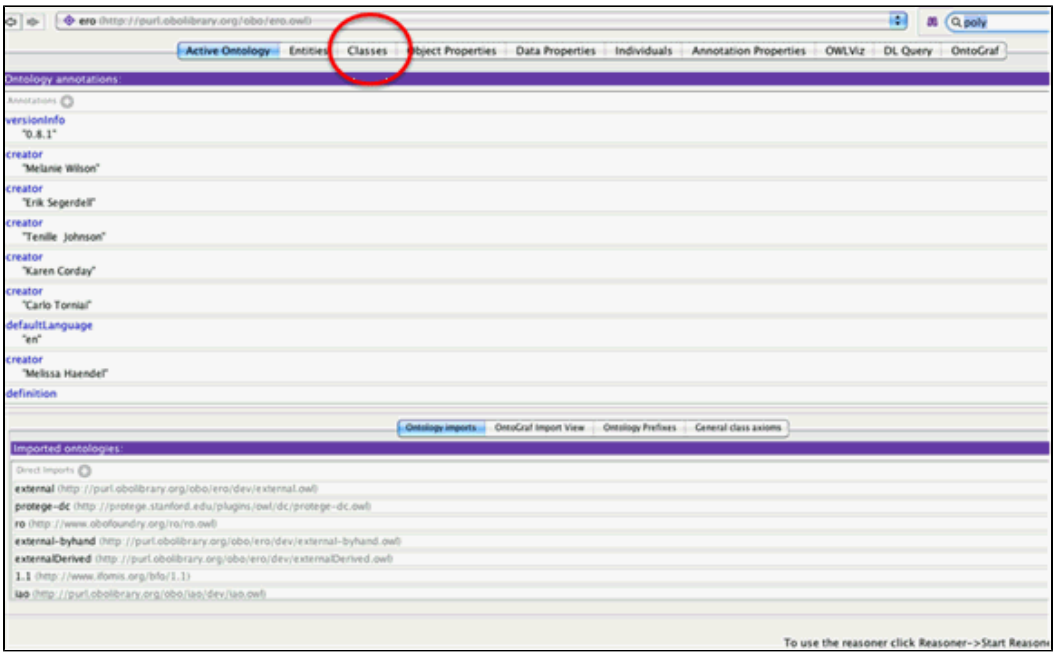
Step 3: Add the new term to the ontology.

Open the Excel file, `IDS.xlsx`, which is typically located here: `SVN ---> src -> ontology`. This file consists of five columns of information labeled as follows: column (A) ERO ID #, column (B) Mireot, (C) Label, (D) Class and column (E) URI.

1. Enter the following information in each column:

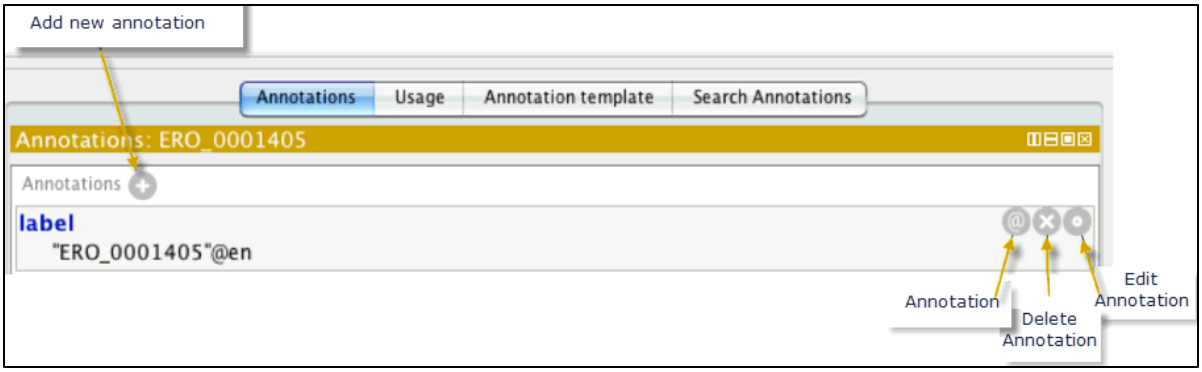
Column A ERO ID #	Column B Mireot	Column C Label	Column D Type	Column E URI
Choose the next consecutive number	Nothing	Enter the name of the term	Typ Class	Choose the next consecutive number

2. Save the file, but leave the spreadsheet open, you'll come back to retrieve the ERO ID#.
3. Open Protégé. Open the OWL ontology. Choose the file, eagle-app.owl from your directory. For example: `/Users/JaneCurator/eagle-i-datamodel/trunk/src/ontology/eagle-i-app.owl.`
4. From the Active Ontology tab, click Classes.



1. The default file that opens is eagle-i-app.owl, use the drop-down menu to change the file to the ero file.
2. If you don't see the parent term, use the search box in the upper-left corner of the screen. In this example, the parent term is technique. Once you find the parent term, highlight the correct class and then click Subclass to add a subclass.
3. When the dialog box, Create a new Owl Class, appears copy and paste the ERO ID (from Column A in the Excel spreadsheet) into the text box: "Please enter a class name". Click OK.

1. Click to change the label for the new term you just added. Highlight the existing label and replace the ERO_xxxx ID (from the Excel spreadsheet) with the name of the term.
   All labels MUST be in lowercase and every annotation must be a datatype, String.
2. To change the datatype, choose String from the Type drop-down list at the bottom of the Annotation box. Click OK.



1. Choose a definition from the list of annotation properties. The definition must be IAO (Information Artifacts Ontology); hover over the definition to make sure. If necessary, you can add an alternate term. Click to add a new annotation. Again, make sure the type is STRING. Click OK.
2. Add two annotation values: definition editor and definition source: Click (add new annotation) and select definition editor from the annotation properties list and type your name using the format: PERSON: First name Last name. Click OK.
3. Click again, and select definition source.
4. Copy and paste the definition source. Make sure the type is STRING. Click OK.
5. Save the term. Click Save and hover over the term to confirm it has a valid URI.

Step 4: Change the term label to an eagle-i preferred label.

Follow these next series of steps to the eagle-i preferred label. Typically, new terms entered into an ontology are always lowercase. Exceptions are made for terms that are proper names or acronyms, like DNA. The eagle-i preferred label is the proper label for the term, typically with the first letter capitalized. For example, polymerase chain reaction would be entered as "polymerase chain reaction" in the `ero.owl` file and the eagle-i preferred label would be Polymerase chain reaction.

1. From the toolbar, switch to eagle-i-app.owl.
2. Use the search box to locate the class you wish to re-label.
   Note: Terms that are in bold are from the currently open ontology. Terms that are not bold, were imported from another file (ontology).
3. Click to add a new annotation value.
4. From the list, select the eagle-i preferred label. Note: eagle-i preferred label should be bold. If it is not bold, check to make sure you are in the correct file.
5. Click OK.
6. Type the name/label (Use capitalization as appropriate, typically capitalize the first letter of the word)
7. Click OK and then click Save.

Step 5: Use the Hermit Reasoner to classify the ontology and make sure it is consistent.

1. From the toolbar, select Reasoner > Hermit 1.3.2.
2. Select Start Reasoner.
3. If the reasoner finds errors, correct the errors and run it again. If no errors are found, move to the next step.

Step 6: Commit the changes in Subversion

Best practice is to make only a few changes before committing to SVN. Keep detailed notes of the changes you make and always add comments describing the hanges you've made when committing.

1. Open the SmartSVN Professional, click Refresh.
2. Select CHANGES. Confirm that everything looks okay and there were no mistakes. Here are two examples of changes in SVN. Often times, SVN will display changes that are not actual changes but just mirror images of terms. Look carefully at the changes and make sure they are either just mirror images or a term that was inserted.
3. Select changed files, (files with a red icon). Click Commit. Make sure to enter comments.

Step: 7: Notify the curators that the file has been unlocked

This is the last step. Notify the curators that the file has been unlocked using an e-mail that looks similar to: `[UNLOCKING] ero.owl and eagle-i-app.owl.`

Obsoleting a Term in the Ontology

Before obsoleting a term it is best practice to check the usage of the terms across the repositories. You can do this using either SPARQL queries or ECDT tools.

Step 1: Update Subversion

1. Open SmartSVN. The files you will be working with are located: svn+ssh://orchestra.med.harvard.edu/svn/cbmi/u24/eagle-i-dev/datamodel/trunk /src/ontology
2. Click Update to update the entire trunk.
   This will update three files: `ids.xlsx, ero.owl` and `eagle-i-app.owl`, as well as all relevant imports.

Step 2: Obsolete the term

1. Open Protégé Open the OWL ontology.
2. Choose the file, eagle-app.owl from your directory. For example:
   `/Users/JaneCurator/eagle-i-datamodel/trunk/src/ontology/eagle-i-app.owl.`
3. From the Active Ontology tab, click Classes.
   The default file that opens is the eagle-i-app.owl file, use the drop-down menu to change the file to the ero file.
4. If you don't see the term, use the search box in the upper-left corner of the screen to find the term.
5. To find out if the term you want to obsolete is used as domain and range for some property or some logical definition, select the term and click Usage. If it is, you will need to replace the obsolete class with the new class at the end of the process or delete the properties and logical definitions that will be obsoleted along with the resource.
6. Rename rdfs:label to obsolete_rdfs:label. Look at a pre-existing obsolete term for an example.
7. Click to add the annotation property, "has obsolescence reason" and type the reason why this term is being obsoleted. If it is being replaced by a new term, enter the new term.
8. Add new parent term: In the description box, under Superclass, click
9. Delete the existing parent name and replace with `ObsoleteClass`. Click OK.
10. Look under the obsolete term you obsoleted, and recheck the usage of the obsoleted class (step #6) and fix any property or logical definitions that still include the obsoleted resources.

Remember to keep track of any data that is currently using the obsoleted term. That data will need to be migrated. Provide a migration (Excel) file for the build team.

Step 3: Use the Hermit Reasoner to classify the ontology

1. From the menu, select Reasoning > Hermit 1.3.2.
2. Select Start Reasoner.

Step 4: Commi to SVN

1. Open SmartSVN.
2. Click Update to check for any changes.
3. Write a detailed list of the changes that have been made.
4. Commit.

Annotation Properties

Application-specific properties and annotations are kept in the eagle-i app file. https://sites.google.com/a/eagle-i.org/workspace/build-team-workspace/data-model-integration/applicationspecificannotations for most current values) The purpose of these annotation properties and their values is to allow curators to "flag" classes and properties to "tell" the eagle-i, for example, to which class is a resource is connected or which property connects a resource to a lab.

Currently the following properties that can be used:

{{- inClassGroup

- inPropertyGroup}}

There is a specific set of values for both

{{inClassGroup and

inPropertyGroup}}

(see https://sites.google.com/a/eagle-i.org/workspace/build-team-workspace/data-model-integration/applicationspecificannotations for most current values)

You can see all these values in the ontology by:

1)Selecting the eagle-i-app-ont ontology as the native ontology in Protégé from the drop-down of active ontology as shown below:

2)Click on the individuals tab and check each single value (note that there is comments explaining the usage and i the Types" indicates if value applies to classes or properties

In order to flag a class or properties with a particular value:

1)Make sure you are adding annotation in the eagle-i-app ontology (see below)

2)Select the class or property form the class or property hierarchy that you want to annotate

3)Select the annotation tab on the right panel and click on add annotation

4)Select inClassGroup or inPropertyGroup from the list (depending what you are annotating)

5)Select 'Individuals' and 'Entity URI' form the right tabs as shown below:

6)Select the value of annotation from the list. And click OK.

For the specifics of the values and meaning of current annotation values please refer to:

[https://sites.google.com/a/eagle-i.org/workspace/build-team-workspace/data-model-integration/applicationspecificannotations
]