The OWL of Biomedical Investigations

Mélanie Courtot [a,*], William Bug [b], Frank Gibson [c], Allyson L. Lister [d], James Malone [e], Daniel Schober [e,f], Ryan R. Brinkman [a], Alan Ruttenberg [g,*]

[a] *Terry Fox Laboratory, British Columbia Cancer Agency, Vancouver, BC, Canada*

[b] *National Center for Microscopy Imaging Research, UCSD, CA, USA*

[c] *School of Computing Science, Newcastle University, Newcastle upon Tyne, UK*

[d] *CISBAN and School of Computing Science, Newcastle University, Newcastle upon Tyne, UK*

[e] *The European Bioinformatics Institute, Cambridge, CB101SD, UK*

[f] *Institute of Medical Biometry and Medical Informatics (IMBI), University Medical Center, 70104 Freiburg, Germany*

[g] *Science Commons, Cambridge, MA, USA*

Abstract

The Ontology for Biomedical Investigations (OBI), written in OWL DL, brings together a large consortium seeking to provide a cross-domain, shared framework for representing investigations in the biological and biomedical sciences. In this paper we report our experiences and describe our development process as it pertains to the implementation in OWL. This includes a number of elements that might inform tool developers as well as suggest general development patterns. Finally, we review where improvements to the OWL format and corresponding tools might be beneficial.

Key words: Ontology; Collaborative development; OWL; Reasoning; Bioinformatics; Ontology for Biomedical Investigations

* Corresponding authors.
Email addresses: mcourtot@gmail.com (Mélanie Courtot), alanruttenberg@gmail.com (Alan Ruttenberg).

1 Introduction

The Ontology for Biomedical Investigations (OBI) Consortium [1] is developing an ontology for the description of biological and clinical investigations. It is written in OWL DL, and built using the Basic Formal Ontology(BFO)[7] as its upper level ontology. The OBI Consortium is a member of the OBO Foundry[11], a collective of developers of science-based ontologies who are establishing a set of principles for ontology development with the goal of creating a suite of interoperable reference ontologies in the biomedical domain. In order to enable development of OBI as a large collaborative project, a strategy was required that would allow concurrent editing, distributed development, version control, offline development, use of different tools and editors, and script-based augmentation of the ontology content. A review of the existing collaborative ontology development tools failed to identify a single application that met these requirements. As a result, we chose to rely on a small group of tools, augmented with a structured mechanism for development; the ontology structure was separated into 10 branches (or sections) for concurrent development by corresponding groups, with each group working more or less independently. This presented some challenges preparing OBI for distribution, as parallel editing of OWL files can lead to non-unique class identifier assignment and conflicts within the ontology. We describe some of these technical challenges and solutions in this paper. One of the fundamental principles of the OBO Foundry is to reuse, where sensible, existing ontology resources. While OWL provides a mechanism to import ontologies (*owl:imports*), this mechanism was not always suitable for OBI. Firstly, current editing tools are not effective for working with very large ontologies such as the NCBI Taxonomy[14] or the Foundational Model of Anatomy[6], making direct OWL imports of such ontologies, as a whole, impractical for day-to-day development. Secondly, other ontologies used by OBI are under active development and may not be aligned with its design (e.g., not using BFO as an upper ontology, or not using OWL DL). Importing such ontologies as a whole would lead to inconsistencies and unintended inferences. One alternative to the OWL built-in import mechanism would be to copy only parts of the external ontology into *obi.owl*. Software that extracts a "module"[5] is in early development and we plan to try such strategies in the future. Our solution was to develop a set of steps or guidelines that describes how to import selected terms without the overhead of importing the complete ontology from which the terms derive. These guidelines are called the Minimum Information to Reference an External Ontology Term (MIREOT) and are described in this paper.

[1] http://purl.obofoundry.org/obo/obi

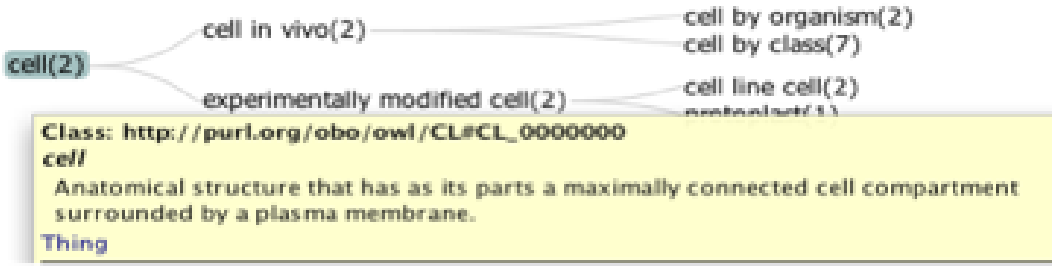

Fig. 1. The to-be-imported *cell* term, as viewed in its original context in the Cell Type ontology class tree. The *cell by organism* and *cell by class* are examples of those we would prefer to not import into OBI.

2 OBI development practices

2.1 Minimum Information to Reference External Ontology Terms (MIREOT)

In deciding upon a minimum unit of import, our first step was to consider the practices of other ontologies. The practice of the Gene Ontology (GO)[2] is that terms are deprecated if their intended meaning changes[12]. We therefore consider terms (*i.e.*, classes) a basic unit of import. According to the MIREOT guidelines the minimum amount of information needed to reference an external class is (i) The source ontology URI; (ii) the source term URI, which generally remains stable and can be used as an unambiguous reference; and (iii) the direct superclass in the target hierarchy (*i.e.*, what OBI class the imported class is a subclass of). We store this minimum information set in a separate file called *external.owl*. Additional information, such as labels and definitions, are mapped into the corresponding OBI annotation properties and stored in a separate file, called *externalDerived.owl*. This additional information is more prone to change, and a distinct file allows for easy update of this information. Updates are managed using a set of templated SPARQL queries [2], which specifies which extra information about the classes in the *external.owl* file to gather. These are then retrieved using queries against the Neurocommons [3] SPARQL [4] endpoint [5].

Figure 1 illustrates how the implementation of the MIREOT guidelines works, with the example of replacing of the OBI class *cell* with that from the OBO Foundry Cell Type (CL) ontology[3]. A second example presents a slightly more complicated challenge. OBI currently uses the NCBI taxonomy for its

[2] http://purl.obofoundry.org/obo/obi/repository/trunk/src/tools/build/external-templates.txt
[3] http://neurocommons.org/
[4] http://www.w3.org/TR/rdf-sparql-query/
[5] http://sparql.neurocommons.org/sparql

species terms. When importing NCBI taxonomy terms as OBI classes, we decided that the information about the term itself was not sufficient on its own: for example if we want to import the term *Mus musculus*, we also want to import its rank information (*e.g.*, genus, kingdom, phylum. to allow for expanded queries. In this case the SPARQL query retrieves all direct superclasses up to one of a set of top-level classes in the taxonomy. Correct inference within OBI and using the external classes is only guaranteed if the full ontologies are imported. We expect to provide an option in the OBI distribution where the *external.owl* file is replaced with a set of corresponding import statements in the *obi.owl* file directly. The import statements are generated by extracting the ontology URIs mentioned in *external.owl*. An important consideration when importing external classes into OBI with this approach is the correctness of OBI assertions made on external terms. In adding new axioms, such as the subclass axiom, to an external term within OBI, the aim is to only assert true statements about the terms. It is likely that an interaction with the external ontology efforts is necessary to collaborate on this important issue. Further, it is also likely that this approach is mutually beneficial in that new knowledge may be gained by the external, source ontologies, whilst OBI has a mechanism for validating this additional knowledge. This should be considered a propitious emergent feature of the collaborative nature of OBO Foundry ontology development and not a disadvantage of the approach.

2.2 Releasing OBI

We required a mechanism that would allow the release of a public version of OBI [6] on a monthly basis. Such a process allows users to acquire a traceable version of the ontology that is essentially static and act as a stable reference point for usage, much like the process undertaken in the software industry. Constructing a single OWL file that contained the entire ontology seemed an appropriate step towards this aim. Having a dedicated release process further allows us to more carefully control and modify the ontology before making it available. Our release process involves a number of automated steps including checks for content quality (*e.g.*, annotations compliant with our policy, OBI-compliant identifiers, disjoints management), syntax (*e.g.*, OWL species validation) and reporting candidate release status to the ontology developers.

[6] The latest version of OBI is always available at http://purl.obofoundry.org/obo/obi.owl

2.3 Quality checks and reports

To ease curators' work whilst ensuring the quality of the ontology, we decided to provide reports to each branch that identified areas not compliant with our policies prior to each release. We use a Jena-based[4] script to read in our branch files and identify missing elements, duplicates or misuse of any of our metadata properties. Such curator reports are rated according to what action needs to be taken: simple warnings for those errors that can be corrected automatically by script or critical alerts for those issues requiring manual intervention from one of our curators. Reports are simple HTML pages displaying terms and associated issues. We explored different policies regarding what to do in case of significant errors (*e.g.*, block release), but instead adopted a release early, release often approach in the hope that this would encourage developers to correct mistakes in a timely fashion. For example, we would occasionally encounter a problem with the Protégé[1] editor with one of our annotation properties being saved in the wrong branch file. For example, when adding a label to one of the instruments after serialization, this label could get stored in the *Biomaterial.owl* file instead of the *InstrumentAndParts.owl* file. As only one file can be edited at a time using Protégé this causes editing problems if the class needs to be updated again later on. In order to mitigate this, we are considering using an extra annotation property to indicate which branch classes belong in. By using this information we could automatically clean up and reorganize branch files according to their logical content. Additional scripts perform other quality control checks, including notification on terms that lack a curation status assignment, listing terms with extra curation instances (*i.e.*, only one is allowed per term), listing terms missing a label, and listing classes that are asserted under a defined class.

2.4 Distributing OBI with inferred superclasses

OBI aims to follow Alan Rector's[9] normalization recommendation by providing an untangled disjoint tree of defined classes and avoiding asserted multiple parenthood on the developers side. However, we also want to provide an enriched post-reasoned user-friendly file. To achieve this we classify the hierarchy and add the inferred superclasses via script to our OWL file, thus allowing end users to have the fully inferred hierarchy, while keeping the curators version of the ontology untangled and "clean".

```
Namespace(e = <http://example.com/>)
Ontology(<http://example.com/>
 Class(e:manuf_role partial e:role)
 Class(e:role partial)
 Class(e:organization partial)
 Individual(e:Affymetrix type(e:organization))
 ObjectProperty(e:has_role )
 ObjectProperty(e:is_manufactured_by
     range(restriction (e:has_role someValuesFrom(e:manuf_role)))
  Class(e:hg133 partial e:microarray)
  Class(e:hg133 partial
     restriction(e:is_manufactured_by value(e:Affymetrix)))
  Class(e:manufacturer complete
     restriction(e:has_role someValuesFrom(e:manuf_role))))
```

Fig. 2. Abstract syntax for an ontology for which the desired inference is not made.

2.5 Assuming that all classes have instances

In Figure 2, we define a *manufacturer* class, an object property *is manufactured by* with range *manufacturer role*, and add that a specific microarray type is manufactured by an organization *Affymetrix*. We were expecting the reasoner to classify *Affymetrix* as *manufacturer*. However this is not the case unless we explicitly add a *microarray* individual to the ontology. In the framework of BFO, all classes must have instances. Ideally, we would indicate our assumption that all classes have at least one individual to a reasoner and have it compute subsumptions and other inferences on that basis, instead of checking satisfiability by asserting that at least one instance exists, serially, for each class. However the reasoners we use, Pellet[10] and Fact++[13], do not offer this choice, and we decided to script the addition of anonymous individuals of each type named in the ontology as part of our release process. We do this for each leaf class, and before computing the inferred superclasses. Asserting a distinct anonymous individual as member of each leaf class means that the superclasses will also have one member and ensures that the entailments will be reliably computed and that ontologies that are not jointly satisfiable will be detected. We plan to suggest that a similar mechanism is adopted by the OWL versions of all OBO ontologies. We note that this choice is not without problems. OBI, augmented with these 'assumed individuals', becomes more difficult to reason with reliably - we have had problems with both Pellet and Fact++ and are at the moment communicating with the developers of those reasoners to determine the source of the problem. Therefore, we currently use the assumed individuals to compute the inferred class hierarchy, but do not include them in the released version of OBI.

2.6 Increasing the readability of the RDF/XML version of OBI

We use numerical identifiers for all our terms: classes, instances, but also for annotation-, data- and object properties. However, we sometimes need to edit

```
<owl:Class rdf:about="&obo;OBI_0000265"> <!-- report table -->
  <!-- definition editor -->
   <OBI_0000274 xml:lang="en">person:Allyson Lister</OBI_0000274>
  <rdfs:label xml:lang="en">report table</rdfs:label>
  <!-- definition -->
  <OBI_0000291 xml:lang="en">A report table is a report display
element consisting of a matrix of cells laid out in a grid, some set of which are filled with some inf
ormation content</OBI_0000291>
  <rdfs:subClassOf>
    <owl:Class rdf:about="&obo;OBI_0000001"/> <!-- report display element -->
  </rdfs:subClassOf>
</owl:Class>
```

Fig. 3. Example of XML comments used to note what IDs correspond to in RDF/XML serialization.

the OWL RDF/XML directly, which is cumbersome because numeric identifiers (IDs) are not easily remembered by humans. To increase human readability of the owl source code we post-process the RDF/XML and generate human readable XML comments for the released version of the file. We recommend that tool developers offer an option to use some annotation property as an XML comment when serializing OWL (Figure 3).

2.7 OBI terms on the Web

In addition to supplying the OBI ontology as a single file, we are in the stage of prototyping a method for responding with a bounded amount of useful information for each URI naming a term in OBI [7]. In doing so we follow httpRange-14 [8] and use a HTTP response code of 303 with a redirect to RDF/XML describing the term. We do no content negotiation to emphasize that the URI names a single thing. In order to present readable information in web browsers, we use an XSL stylesheet, which is executed by the browser to generate HTML. Choices need to be made as to what the content of the RDF delivered at this URL should be. We chose to make each bundle of RDF delivered at this URL a valid OWL DL ontology by importing the full OBI ontology. A certain amount of relevant information is included for web clients that do not follow that import statement: for a class, the axioms defining it, inferred superclasses, properties that it is in the domain of or range of (class usage), and labels for any referenced terms are added. We also include project information using the DOAP schema [9] including pointers to our repository, tracker, mailing list, and release information.

[7] For an example view `http://purl.obofoundry.org/obo/OBI_0000225` in a browser

[8] `http://www.w3.org/2001/tag/issues#httpRange-14`

[9] `http://trac.usefulinc.com/doap`

3 Discussion

Support for MIREOT and for reasoning services that assume the existence of an instance of each class, as discussed above, would be helpful. Below we discuss a number of additional features of OWL and associated tools that would ease development of OBI and other ontology projects.

3.1 Deprecation

As the obsolete terms are stored in a separate file (to make it easier to excise them from some versions of OBI), we constantly run into issues trying to move obsolete classes to the *ObsoleteClass* hierarchy. Protégé allows for editing of a single ontology file at a time (the *active ontology*), making this difficult and error prone. In addition, our deprecation policy stipulates, among other things, that axioms involving deprecated terms should be removed. In order to support this practice we believe it is desirable to have either better tool support for this or OWL language support that would cause axioms involving deprecated/obsoleted terms to be considered annotations.

3.2 Annotations on annotations

OBI is used in a variety of fields and we need to address the fact that one term can mean different things in different communities. For example, the term *probe* is a synonym for the term *reporter* in one community, whereas it is a synonym for the term *detector* in another. Annotations on annotations are an appropriate representation of these community-specific labels, and would allow us to tag any of our synonyms with extra information noting pertinence to a specific community.

3.3 Versioning

OBI's policy is to release frequent updates and to maintain access to all versions. We create dated versions of each release, currently using the Persistent Uniform Resource Locator (PURL)[8] system to provide time robust access to successive revisions. This leaves control over the choice between preferring stability and being up to date with the latest developments to the end-user.

OWL 2's versionURIs [10] will make it possible for users to easily choose which version of the ontology to use. We believe it would be beneficial to the community if this practice became the generally accepted way of managing versioning of ontologies as it provides a single and unambiguous path to accessing version numbers (independent of tools). While developing OBI we prefer stability (*i.e.*, not being surprised by unplanned-for changes), and we work around the lack of published ontology versions by relying on local copies of imported ontologies.

3.4 Support for Rector-normalization style editing

The dominant paradigm for editing ontologies is that of a single rooted hierarchy. However the style proposed by Rector and others is to develop a series of single inheritance ontologies and a separate set of classes defined in terms of elements of the disjoint single inheritance trees. An ontology interface that supported fluidly moving between the component trees, the defined classes, and the inferred entangled (composite) view, as well as providing easy access to common patterns for the composite definitions would significantly benefit ours and others efforts.

3.5 Conclusion

OBI is an ambitious project, uniting a large number of collaborators from different biological and biomedical sciences (more than 45 experts representing 18 communities), many of who plan to use OBI in their own projects. Due to the number and distributed location of developers and domain experts, OBIs needs for collaborative ontology development bring new and currently unaddressed challenges at both the organizational and technical levels.

4 Acknowledgements

In memory of our friend and colleague William Bug, Ontological Engineer.

The OBI consortium is (in alphabetical order): Ryan Brinkman, Bill Bug, Helen Causton, Kevin Clancy, Christian Cocos, Mélanie Courtot, Eric Deutsch, Liju Fan, Dawn Field, Jennifer Fostel, Gilberto Fragoso, Frank Gibson, Tanya Gray, Jason Greenbaum, Pierre Grenon, Jeff Grethe, Mervi Heiskanen, Tina Hernandez-Boussard, Allyson Lister, James Malone, Elisabetta Manduchi,

[10] `http://www.w3.org/2007/OWL/wiki/Syntax#Ontology_URI_and_Version_URI`

Luisa Montecchi, Norman Morrison, Chris Mungall, Helen Parkinson, Bjoern Peters, Matthew Pocock, Philippe Rocca-Serra, Daniel Rubin, Alan Ruttenberg, Susanna-Assunta Sansone, Richard Scheuermann, Daniel Schober, Barry Smith, Holger Stenzhorn, Chris Stoeckert, Chris Taylor, John Westbrook, Joe White, Trish Whetzel, Stefan Wiemann. The authors work is partially supported by funding from the National Institute of Biomedical Imaging and Bioengineering, NIH; Grant Number: EB005034, the EC EMERALD project(LSHG-CT-2006-037686), the BBSRC(BB/C008200/1), the EU NoE NuGO(NoE 503630), the CARMEN project EPSRC(EP/E002331/1), the EU FP7 DebugIT project (ICT-2007.5.2-217139) and the Michael Smith Foundation for Health Research.

References

[1] The Protégé Ontology Editor and Knowledge Acquisition System, `http://protege.stanford.edu`,.

[2] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, G. Sherlock, Gene Ontology: tool for the unification of biology, Nat Genet 25 (1) (2000) 25–29.

[3] J. Bard, S. Rhee, M. Ashburner, An ontology for cell types, Genome Biology 6 (2) (2005) R21.

[4] J. J. Carroll, I. Dickinson, C. Dollin, A. Seaborne, K. Wilkinson, D. Reynolds, D. Reynolds, Jena: Implementing the semantic web recommendations, 2004.

[5] B. Cuenca Grau, I. Horrocks, Y. Kazakov, U. Sattler, Extracting Modules from Ontologies: A Logic-based Approach, in: H. Stuckenschmidt, S. Spaccapietra (eds.), Ontology Modularization, Springer, 2008.

[6] C. Golbreich, S. Zhang, O. Bodenreider, The foundational model of anatomy in OWL: Experience and perspectives, Web Semant. 4 (3) (2006) 181–195.

[7] P. Grenon, B. Smith, L. Goldberg, Biodynamic Ontology: Applying BFO in the Biomedical Domain, in: Stud. Health Technol. Inform, IOS Press, 2004.

[8] E. J. KE Shafer, SL Weibel, The PURL Project, Journal of Library Administration.

[9] A. L. Rector, Modularisation of domain ontologies implemented in description logics and related formalisms including OWL, in: K-CAP '03: Proceedings of the 2nd international conference on Knowledge capture, ACM, New York, NY, USA, 2003.

[10] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, Web Semantics: Science, Services and Agents on the World Wide Web 5 (2) (2007) 51.

[11] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall, O. Consortium, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S. A. Sansone, R. H. Scheuermann, N. Shah, P. L. Whetzel, S. Lewis, The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration, Nature biotechnology 25 (11) (2007) 1251–1255.

[12] The Gene Ontology Consortium, The Gene Ontology editorial guide, `http://www.geneontology.org/GO.usage.shtml`.

[13] D. Tsarkov, I. Horrocks, FaCT++ description logic reasoner: System description, in: Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006), vol. 4130 of Lecture Notes in Artificial Intelligence, Springer, 2006.

[14] D. L. Wheeler, T. Barrett, D. A. Benson, S. H. Bryant, K. Canese, D. M. Church, M. DiCuccio, R. Edgar, S. Federhen, W. Helmberg, D. L. Kenton, O. Khovayko, D. J. Lipman, T. L. Madden, D. R. Maglott, J. Ostell, J. U. Pontius, K. D. Pruitt, G. D. Schuler, L. M. Schriml, E. Sequeira, S. T. Sherry, K. Sirotkin, G. Starchenko, T. O. Suzek, R. Tatusov, T. A. Tatusova, L. Wagner, E. Yaschenko, Database resources of the National Center for Biotechnology Information, Nucleic acids research 33 (Database issue) (2005) D39–45.