Applied Ontology 6 (2011) 13–22
DOI 10.3233/AO-2011-0086
IOS Press

Overcoming the ontology enrichment bottleneck with Quick Term Templates

Philippe Rocca-Serra [a,*], Alan Ruttenberg [b], Martin J. O'Connor [c], Patricia L. Whetzel [c], Daniel Schober [d], Jay Greenbaum [e], Mélanie Courtot [f], Ryan R. Brinkman [f], Susanna Assunta Sansone [a], Richard Scheuermann [g], The OBI Consortium and Bjoern Peters [e]
[a] *Oxford e-Research Centre, University of Oxford, Oxford, UK*
E-mails: proccaserra@gmail.com, sa.sansone@gmail.com
[b] *Science Commons, Cambridge, MA, USA*
E-mail: alanruttenberg@gmail.com
[c] *Stanford University, Stanford, CA, USA*
E-mails: martin.oconnor@stanford.edu, whetzel@stanford.edu
[d] *Institute for Medical Biometry and Medical Informatics, University Clinic, Freiburg, Germany*
E-mail: schober@imbi.uni-freiburg.de
[e] *La Jolla Institute for Allergy and Immunology, La Jolla, CA, USA*
E-mails: bpeters@liai.org, jgbaum@gmail.com
[f] *Terry Fox Laboratory, British Columbia Cancer Agency, Vancouver, BC, Canada*
E-mails: mcourtot@gmail.com, rbrinkman@bccrc.ca
[g] *Department of Pathology and Division of Biomedical Informatics, U.T. Southwestern Medical Center, Dallas, TX, USA*
E-mail: richard.scheuermann@utsouthwestern.edu

Abstract. When developing the Ontology of Biomedical Investigations (OBI), the process of adding classes with similar patterns of logical definition is time consuming, error prone, and requires an editor to have some expertise in OWL. Moreover, the process is poorly suited for a large number of domain experts who have limited experience with ontology development, and this can hinder contributions. We have developed a procedure to ease this task and allow such domain experts to add terms to the ontology in a way that both effectively includes complex logical definitions, yet requires minimal manual intervention by the OBI developers. The procedure is based on editing a Quick Term Template in a spreadsheet format that is subsequently converted into an OWL file. This procedure promises to be a robust and scalable approach for ontology enrichment as evidenced by encouraging results obtained when evaluated with an early version of the MappingMaster Protégé plugin.

Keywords: Quick Term Template, ontology coverage, Ontology for Biomedical Investigations, pattern, MappingMaster Protégé plugin

1. Introduction

The Ontology for Biomedical Investigations (The OBI Consortium, 2009) consortium is developing an integrated ontology for the description of life science and clinical investigations based on the re-

[*] Corresponding author: Philippe Rocca-Serra, Oxford e-Research Centre, University of Oxford, 7 Keble Road, OX1 3QG, Oxford, UK. Tel.: +44 01865 610 657; Fax: +44 01865 610 612; E-mail: proccaserra@gmail.com.

quirements of a diverse set of scientific communities. Briefly, OBI's development process is as follows: candidate terms for inclusion in the ontology are provided from case studies and user requests. Textual and formal definitions for the candidate terms are developed, taking into account the class categorization. A variety of techniques, practiced at frequent teleconferences, biannual meetings and by individual developers is used to develop definitions. Those definitions are then categorized along the three main axes provided by the Basic Formal Ontology (BFO) (Grenon et al., 2004) on which OBI is built. Where possible, OBI reuses classes already defined, or which are in the scope of partner ontologies from the OBO foundry (Smith et al., 2007). These practices help ensure that classes are placed in a correct *is_a* hierarchy, and that OBI does not redundantly define entities.

In order to represent more complex relationships among entities, the Web Ontology Language (OWL) (Smith et al., 2004) is used to express axioms that relate entities used in OBI. By creating classes with logically necessary and sufficient definitions, a complex, expressive, and logically rigorous domain representation is constructed that can be practically maintained and validated by reasoners, such as Pellet (Sirin et al., 2007) and FaCT++ (Tsarkov & Horrocks, 2006). However, as experienced by OBI developers, manually adding classes and logical axioms remains a time-consuming (Bodenreider & Stevens, 2006), possibly error prone process, in spite of using advanced ontology editors such as Protégé (The Protégé Ontology Editor and Knowledge Acquisition System, 2009). Also, using such editor assumes knowledge of OWL, thus significantly limits the number of people who can contribute productively to enriching the ontology.

Considering that many hundreds of candidate terms are being submitted to OBI, the process of defining them must not become a bottleneck. Our proposal is motivated by the observation that definitions of a significant proportion of term requests can be accommodated by a limited number of pre-defined design patterns. The approach we describe here falls into the realm of content ontology design pattern (CP), a subgroup of design patterns used in ontology engineering. Ontology design patterns (ODP) cover those techniques used to solve common and recurring representation problems. The focus of the present manuscript is not to provide a review of such work, for which extensive and relevant literature exists (Aranguren et al., 2008; Aroyo, 2009; Corcho et al., 2009; Iannone et al., 2009). Rather, it insists on presenting a practical solution that is geared toward bioontologies developers and editors. In order to engage domain experts without extensive practice in ontology development, we formulate the required input for each such design pattern as a *Quick Term Template* (QTT), which can be edited as an Excel spreadsheet.

In the following, we illustrate an example of a common term request: assays that measure the concentration of a specified molecular compound in a given material, which is typical for clinical chemistry assays. Requests for terms to identify such assays come from diverse communities, including the BioInvestigation Index (The Bio Investigation Index, 2009; Field et al., 2009), the Immune Epitope Database (Peters & Sette, 2007) and the Influenza Virus BioHealthbase (Squires et al., 2008). This example illustrates the QTT process as a proof of principle. We claim that the approach can be easily extended to different design patterns.

2. Methodology and results

The QTT submission process has four main steps: 1. agreement by The OBI Consortium on the logical definition of the parent class for submissions matching a certain pattern; 2. identification of entities that can be varied with respect to the parent class (the differentia), for which a QTT spreadsheet containing one column for each such entity is generated; 3. population of a QTT spreadsheet by domain experts; 4. processing the QTT submission to generate new classes and definitions, and returning the label and identifier of an OBI class for each valid row of the submission.

Step 1: *Develop the representation of the parent class*

The example used throughout this section is a QTT for subclasses of '*analyte assay*' in OBI (OBI:0000443[1]). Assays of this type measure the concentration of a specified molecular entity relative to a given material entity, such as *measuring glucose concentration in blood in units of* μg *per liter*. Each logical definition relates the material in which the concentration is measured (the evaluant; e.g., blood), the molecular entity that is detected (the analyte; e.g., glucose), and the units of the measurement being made (e.g., microgram per liter). The full logical definition of this class is shown in Fig. 1. The corresponding textual definition is: "An *analyte assay* is an assay with the objective to determine the concentration of one substance (bearer of the analyte role) that is present in (part of) another (bearer of the evaluant role)". The output of the assay is information about concentration – a relational quality of the analyte towards the evaluant.

Step 2: *Derive tabular Quick Term Template*

We found that a large number of our current requests for terms are subclasses of analyte assay. Their differentiae are the analyte (i.e., what the concentration is being detected of), the evaluant (i.e., the material in which the analyte concentration is detected), and the unit, which is used to qualify the measurement datum. Consequently, a QTT for an analyte assay needs columns for only those three entities. Table 1 depicts a QTT with several example entities as they would be seen in a spreadsheet by a submitter. Each column is to be filled with elements that are of a specified general type. The analyte column is expected to be a subclass of molecular entity, and the evaluant column any material entity.

```
Analyte assay:
'achieves planned objective' some 'analyte measurement objective'
   and realizes some ('evaluant role' and ('role of' some 'material entity'))
   and realizes some ('analyte role' and
                            ('role of' some ('scattered molecular aggregate' and
                                                        'has grain' only 'molecular entity')))
   and has_specified_output
         some ('scalar measurement datum'
                    and ('is quality measurement of' some 'molecular concentration')
                           and ('has measurement unit label' some 'concentration unit label'))
```

Fig. 1. OWL restrictions that logically define the analyte assay class in OBI. The three underlined classes are the differentia varied in the QTT template in order to construct more specific subclasses.

[1]Here and throughout the paper identifiers are abbreviated URIs. URI prefixes are: OBI: http://purl.obolibrary.org/obo/OBI_, CHEBI: http://purl.org/obo/owl/CHEBI#CHEBI_, UO: http://purl.org/obo/owl/UO#UO_, PRO: http://purl.obolibrary.org/obo/PRO_, FMA: http://purl.org/obo/FMA#FMA_, BFO: http://www.ifomis.org/bfo/1.1/snap#.

Table 1

A basic QTT for submitting an analyte assay term request

Analyte label	Analyte ID	Evaluant label	Evaluant ID	Measurement unit label	Measurement unit ID
Glucose	CHEBI:17234	Portion of blood	FMA:9670	mmol per liter	UO:0000300
Sodium chloride	CHEBI:26710	Blood plasma	OBI:0100016	mmol per liter	UO:0000300
Chromium-51	CHEBI:50076	Cell culture supernatant	OBI:1000023	ppm	UO:0000169
Glucose	CHEBI:17234	Material_entity	BFO:MaterialEntity	mmol per liter	UO:0000300
Interferon gamma	PRO:00000001	Cell culture supernatant	OBI:1000023	µg per liter	UO:0000301

Notes: This specification includes classes defined in several ontologies: Chemical Entities of Biological Interest (ChEBI) (Degtyarenko et al., 2008), The Foundational Model of Anatomy (FMA) (Rosse & Mejino, 2003), the Unit Ontology (UO) (Gkhoutos, 2005), the Protein Ontology (PRO) (Natale et al., 2007) and BFO.

Step 3: *Domain experts populate the template*

The template hides the complexity of modeling by only identifying the differentiating entities needed for the definition of the class while hiding the actual relations binding those entities together. The burden of adding logical definitions is displaced from the users to the machine, which reliably and automatically populates class specifications from the template and the differentiating entities supplied by the user. A template such as this one would be accompanied by guidelines for users explaining what values are allowed in the columns, and how they will be interpreted in building the assay.

Step 4: *Submission processing*

After submission, a QTT processing goes through the steps outlined below. For brevity we omit a discussion of error handling.

Step 4.1. Identify referenced classes from external ontologies, and import them as necessary via the MIREOT mechanism (Courtot et al., 2009). OBI relies on this mechanism to reference classes in external ontologies. In case entities are absent from resources, a submission is necessary. This may be perceived as a hindrance, but based on our experience the processing of term request was quick enough not to be perceived as a hindrance to the process.

Step 4.2. Create an OWL class description by substituting values from the spreadsheet.

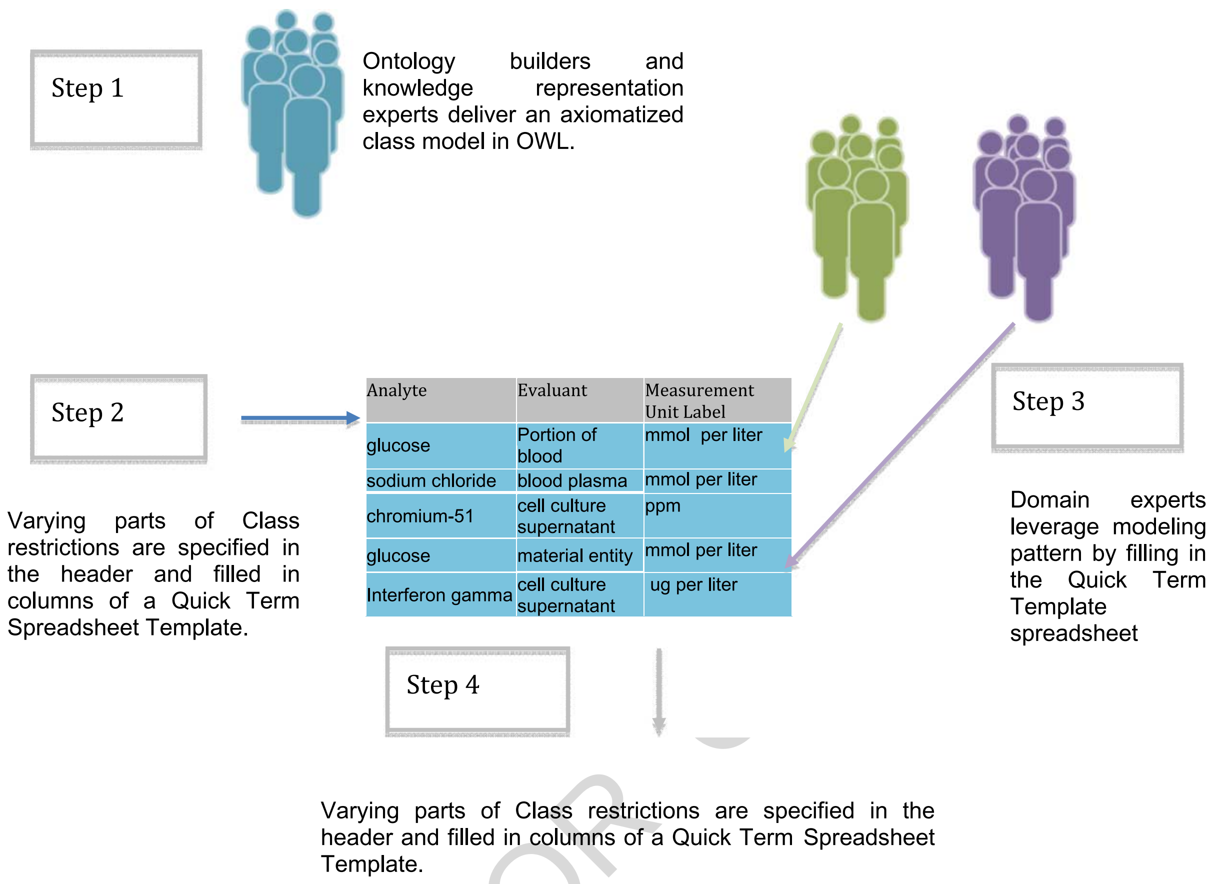
Step 4.3. Use the constructed class description to do a query for equivalent classes already in OBI. If an equivalent class exists, store its URI and label and continue to the next row.

Step 4.4. If an equivalent class does not already exist in OBI, create a unique OBI URI and associate with it a new class defined by the constructed class description. Add metadata such as label and definition. Since a QTT submission creates fully logically defined classes, the creation of labels and textual definitions can be automated. In the Table 1 examples, the class defined by row 1 is assigned the label ‘glucose concentration measurement in blood in units of mmol per liter’, the class corresponding to row 5 is assigned the label ‘interferon gamma concentration measurement in cell culture supernatant in units of µg per liter’.

Step 4.5. Use a reasoner to perform a consistency check and classification on entities found both in the OBI core file and the QTT output file which holds the newly created classes.

Step 4.6. Report on processing of the template, and return a list of URIs and labels corresponding to the rows of the submitted QTT spreadsheet.

An overview of the process is presented in Fig. 2.


Step 1
Ontology builders and knowledge representation experts deliver an axiomatized class model in OWL.
Step 2
Varying parts of Class restrictions are specified in the header and filled in columns of a Quick Term Spreadsheet Template.
Step 3
Domain experts leverage modeling pattern by filling in the Quick Term Template spreadsheet
Step 4
Varying parts of Class restrictions are specified in the header and filled in columns of a Quick Term Spreadsheet Template.

Analyte	Evaluant	Measurement Unit Label
glucose	Portion of blood	mmol per liter
sodium chloride	blood plasma	mmol per liter
chromium-51	cell culture supernatant	ppm
glucose	material entity	mmol per liter
Interferon gamma	cell culture supernatant	ug per liter

Fig. 2. Overview of the process using the OBI modeling of the class 'analyte assay' as a starting point or seed class (step 1). The variable parts (differentiae) of the representation are used to derive a Quick Term Template made up of 3 fields (step 2 and 3). The OWL file is generated using a dedicated tool (step 4). For simplicity, identifiers were omitted in this figure. (Colors are visible in the online version of the article; http://dx.doi.org/10.3233/AO-2011-0086.)

3. Implementation

The implementation of step 1–3 in the QTT approach requires no automation, and is demonstrated for the 'analyte assay' example in Fig. 1 and Table 1. Step 4 requires implementation of an automated QTT handler. In order to validate the approach, a prototype of a QTT handler was created using Perl. Plain OWL templates were derived from the previously described class representation as found in the ontology and populated with token values parsed out from the incoming QTT spreadsheet template (step 4.5). This prototype implementation delivered the expected results and helped in refining requirements of the workflow for step 4. However, running this implementation required extensive manual intervention making it not end-user friendly. Therefore other options were considered.

As OBI developers are heavy users of the Protégé editor, its plugin library was explored for alternative implementation options. Three add-ons (the Matrix (Drummond, 2008), Excel Import (Kola, 2009) and OPPL plugins for Protégé 4 (Iannone et al., 2009)) were evaluated. The Matrix plugin enables tabular visualization of the axioms of an existing class viewed in Protégé. Potentially, this allows rapid crafting of a QTT from a class. However, the lack of a persistence mechanism for saving such a template significantly limits direct applicability of the Matrix plugin for the QTT approach. The Excel Import plugin has a fairly explicit aim: taking in an Excel spreadsheet and creating OWL classes by relying on a set of rules to declare the relations between columns. This is technically very close to what is required for implementing the QTT specifications. However, while assessing the relevance of the Excel Import plugin, two major stumbling blocks appeared. First, the incoming spreadsheet had to be explicit, meaning that all restrictions and fillers placeholders must be present as column headers for the OWL generation to occur properly. This requirement defeats the purpose of the QTT, which aims to conceal some of the modeling complexity from end users. Second, it is not possible to create nested axioms on a class X, such as "X (*realizes some* ('*evaluant role*' *and* (*role_of some* Y)))", from the tool's Restriction Generator pane. This second limitation means that Excel Import could be used for fairly simple and direct class restrictions, but it is incompatible with some of the more advanced patterns required by OBI developers.

More flexibility in specifying axioms and manipulating OWL ontologies is provided by the OPPL plugin, which relies on the Manchester syntax (Manchester Syntax). However, the OPPL plugin requires Protégé 4 while some of OBI development still relies on Protégé 3.4 features.

All three of these tools are highly useful and fully functional for their intended applications, but each was missing functionality necessary to develop an end-to-end prototype for QTT template processing. Specifically, none provided the ability to do all of the following: build class expression templates of the complexity shown in Fig. 1, persist such templates, and populate them by parsing information from a spreadsheet. Instead we used a prerelease version of MappingMaster, a plugin for Protégé 3.4 (O'Connor et al., 2010) that is under active development. The following section describes our experience with this tool.

MappingMaster is an open source Protégé-OWL version 3.4 plugin under development for mapping spreadsheet content into OWL. It provides a Domain Specific Language (DSL) that is based on the Manchester Syntax to define these mappings. In this DSL, any reference to an OWL named class, OWL property, OWL individual, or a data value can be substituted with a reference to one or more cells within a spreadsheet. Any expressions containing such references are preprocessed and the relevant spreadsheet content specified by these references is imported. This content can then be used in four main ways: (1) It can be used to directly name OWL entities that are created on demand. (2) It can be used to annotate OWL entities that are created on demand. (3) The content may reference existing OWL entities, either directly as a URI or through an annotation. (4) Finally, the content may be used as a literal data value. Using one of these approaches, each reference within an expression is resolved during preprocessing to a named OWL entity or a data value and the resolved value is substituted for its associated reference. A standard Manchester syntax processor can then interpret the resulting expression and generate the OWL equivalent statement.

Declaratively specifying mappings in this way has several advantages. No programming or scripting expertise is required to write the mappings, and they can be easily shared using the MappingMaster plugin where they can be persistently stored as OWL files. The mappings can then easily be executed repeatedly on different spreadsheets with the same structure. Since MappingMaster is available as a Protégé plugin, the results of mapping processing can be examined immediately within the ontology editor, and the mappings modified as needed and immediately re-executed, speeding the development process. MappingMaster also includes an interactive editor for the mapping DSL that supports on-the-fly entity name checking and dynamic expansion of entity references.

To implement a QTT parser capable of creating analyte assays corresponding to the definition shown in Fig. 1, we have put to the test a development version of MappingMaster, available as an add-on to

```
Class: @A*(rdfs:label ‘analyte assay’)
EquivalentTo:
(achieves_planned_objective some ‘analyte measurement objective’) and
(realizes some (‘evaluant role’ and (role_of some @D*(material_entity)))) and
(realizes some (‘analyte role’ and
                (role_of some (‘scattered molecular aggregate’ and
                               (‘has grain’ only @B*(‘molecular entity’))))))
SubClassOf:
has_specified_output some
    (‘scalar measurement datum’ and
      (‘is quality measurement of’ some ‘molecular concentration’) and
       (‘has measurement unit label’ some @F*(‘measurement unit label’)))
```

Fig. 3. Template expressions in MappingMaster’s DSL based on the Manchester Syntax. References to spreadsheet cells are prefixed with “@”. Cell values are substituted into the template by MappingMaster to generate class descriptions associated with the QTT.

Protégé 3.4.2.[2] A ‘How To’ document describes the whole procedure, with screenshots, an example Quick Term Template, and the resulting OWL file (Supplementary File 1). It demonstrates the validity of both the approach and the tool. A completed sample QTT template is available as an Excel document (Supplementary File 2) that contains the information presented in Table 1, and an additional column in which the label for the newly generated class is constructed based on the labels of its differentiae using Excel formulas. The DSL expressions used to convert the QTT template into OWL classes as shown in Fig. 3 are passed to MappingMaster. Running the tool generates a supplemental OWL file that contains all newly created classes (Supplementary File 3). All the material is also available from OBI wiki (Rocca-Serra, 2009).

4. Conclusion and future work

The QTT process outlined here provides two benefits. First, it provides a method to incorporate a large number of classes considered of high value by domain experts in the communities OBI is designed to serve. For example, the IUPAC clinical chemistry resources (International Union of Pure and Applied Chemistry, 2007) contain hundreds of assays describing analyte measurements. Second, the approach allows domain experts to directly populate templates without having to learn OWL syntax.

The evaluation of MappingMaster Protégé plug-in as a QTT handler, has produced encouraging results. This early version already exhibits key features such as flexible creation of axioms (thanks to a domain specific language based on the Manchester Syntax) as well as capabilities to automatically generate names for the newly created defined classes by passing user defined expression to the `rdfs:label` field. Finally, it tries to avoid class duplication by inspecting the target ontology for existing entries matching the input received from the template.

As always with ‘off the shelf’ solutions such as MappingMaster, there are some caveats: some portions of the QTT specifications are not entirely supported and so reaching production grade reliability will require further work. Several rounds of evaluation have led to a number of feature requests, which would facilitate performing the entire procedure.

[2]Protégé extensions can be downloaded from http://protege.stanford.edu/download/download.html.

Three areas would benefit from such efforts. First, we would like to have the capability to perform automatic class resolution when dealing with external ontologies. This would require implementing the MIREOT mechanism. At present, external reference resolution in a QTT template is done manually prior to running MappingMaster, with processing aborted if any term is not found. It should be noted that ISACreator (Rocca-Serra et al., 2010; Field et al., 2009), a spreadsheet editor geared towards managing experimental metadata ships with embedded ontology lookup service, could be harnessed to create and populate Quick Term Template in order to address this limitation.

Second, development could be made more efficient by checking class membership and checking class equivalence as run-time queries rather than relying on full reclassification of the ontology, which can be very time consuming. In the example of analyte assays we have presented, this would allow quick detection of classes declared as analyte (first column in Table 1) but which are not subtypes of 'molecular_entity', as expected. Reporting such errors immediately would speed up debugging of submitted QTT spreadsheets.

Third, we would like better support for adding class metadata. It is currently not possible to create class annotations such as cross-references, editor notes or alternative names, which would be easily supplied when creating the QTT spreadsheet. We are aware that future releases of MappingMaster plugin will cater for this need.

Finally, with the advent of OWL 2 and Protégé 4 to support it, porting MappingMaster to use the OWLAPI used by Protégé 4 would be a natural evolution. Plans are underway to achieve this.

Going forward, we plan to create a library of Quick Term Templates encapsulating definition patterns for any area of OBI for which there are frequent term requests. Documentation and software could augment and refine those templates in order to restrict which entities may be selected in each column to specified source ontologies or even certain subclasses from those ontologies. We believe the QTT approach outlined here will prove useful to all ontology development projects that want domain experts to contribute directly to development while enforcing strict logical definitions.

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, Melanie Courtot, Dirk Derom, Eric Deutsch, Liju Fan, Dawn Field, Jennifer Fostel, Gilberto Fragoso, Frank Gibson, Tanya Gray, Jason Greenbaum, Pierre Grenon, Jeff Grethe, Yongqun He, Mervi Heiskanen, Tina Hernandez-Boussard, Lawrence Hunter, Philip Lord, Allyson Lister, Monnie McGee, James Malone, Elisabetta Manduchi, 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, Larisa Soldatova, Holger Stenzhorn, Chris Stoeckert, Chris Taylor, John Westbrook, Joe White, Trish Whetzel, Stefan Wiemann, Jie Zheng.

We would like to thank the reviewers for their constructive appraisal of our manuscript, and their suggestions and comments for improvement.

This work was supported by funding from NIH grants and contracts HHSN26620040006C, N01AI40041, N01AI40076, UL1RR024982, R01EB005034, U54 HG004028, the Public Health Agency of Canada, Canadian Institutes of Health Research Influenza Research Network (PCIRN), the Michael Smith Foundation for Health Research, EU NoE NuGO (NoE 503630), EU IP CarcinoGenomics (PL 037712), NERC-NEBC, BBSRC's BB/G000638/1, BB/E025080/1, BB/D524283/1 and RC UK.

This work was conducted using the Protégé resource, which is supported by grant LM007885 from the United States National Library of Medicine and NSF STTR Award #0750543 funded the development of MappingMaster.

References

Aranguren, M.E., Antezana, E., Kuiper, M. & Stevens, R. (2008). Ontology design patterns for bio-ontologies: a case study on the cell cycle ontology. *BMC Bioinformatics*, *9*(Suppl. 5), S1.

Aroyo, L. (2009). The semantic web research and applications. In *Proceedings of 6th European Semantic Web Conference, ESWC 2009*, Heraklion, Crete, Greece, May 31–June 4, 2009. Berlin: Springer.

Bodenreider, O. & Stevens, R. (2006). Bio-ontologies: current trends and future directions. *Briefings in Bioinformatics*, *7*(3), 256–274.

Corcho, O., Roussey, C., Manuel Vilches Blázquez, L. & Pérez, I. (2009). Pattern-based OWL ontology debugging guidelines. In *Workshop on Ontology Patterns*. Presented at *8th International Semantic Web Conference*, Washington, DC, USA. Available at: http://ceur-ws.org/Vol-516/pap02.pdf.

Courtot, M., Gibson, F., Lister, A., Malone, J., Schober, D., Brinkman, R. & Ruttenberg, A. (2009). MIREOT: the minimum information to reference an external ontology term. In *International Conference on Biomedical Ontology*, Buffalo, NY, USA.

Degtyarenko, K., de Matos, P., Ennis, M., Hastings, J., Zbinden, M., McNaught, A., Alcántara, R. et al. (2008). ChEBI: a database and ontology for chemical entities of biological interest. *Nucleic Acids Research*, *36*(Database issue), D344–D350.

Drummond, N. (2008). MatrixViews – co-ode-owl-plugins – spreadsheet-style views for your ontology – project hosting on Google code. *MatrixViews – co-ode-owl plugins*. Available at: http://code.google.com/p/co-ode-owl-plugins/wiki/MatrixViews.

Field, D., Sansone, S., Collis, A., Booth, T., Dukes, P., Gregurick, S.K., Kennedy, K. et al. (2009). Megascience. 'Omics data sharing. *Science*, *326*(5950), 234–236.

Gkhoutos, G. (2005). Unit ontology. Available at: http://www.berkeleybop.org/ontologies/owl/UO.

Grenon, P., Smith, B. & Goldberg, L. (2004). Biodynamic ontology: applying BFO in the biomedical domain. *Studies in Health Technology and Informatics*, *102*, 20–38.

Iannone, L., Egaña, M., Rector, A. & Stevens, R. (2009). Augmenting the expressivity of the ontology pre-processor language. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.143.852.

International Union of Pure and Applied Chemistry (2007). Available at: http://old.iupac.org/divisions/VII/VII.C.1/index.html.

Kola, J. (2009). ExcelImport – co-ode-owl-plugins – get data from a spreadsheet into your ontology – project hosting on Google code. Available at: http://code.google.com/p/co-ode-owl-plugins/wiki/ExcelImport.

Natale, D.A., Arighi, C.N., Barker, W.C., Blake, J., Chang, T., Hu, Z., Liu, H. et al. (2007). Framework for a protein ontology. *BMC Bioinformatics*, *8*(Suppl. 9), S1.

O'Connor, M., Halaschek-Wiener, C. & Musen, M. (2010). M2: a language for mapping spreadsheets to OWL. Presented at *OWLED 2010*, San Francisco, CA.

Peters, B. & Sette, A. (2007). Integrating epitope data into the emerging web of biomedical knowledge resources. *Nature Reviews. Immunology*, *7*(6), 485–490.

Rocca-Serra, P. (2009). Quick Term Templates – OBI ontology. Available at: http://obi-ontology.org/page/Quick_Term_Templates#Processing_the_Analyte_Assay_Template_using_Mapping_Master:_the_procedure_from_start_to_finish.

Rocca-Serra, P., Brandizi, M., Maguire, E., Sklyar, N., Taylor, C., Begley, K., Field, D. et al. (2010). ISA software suite: supporting standards-compliant experimental annotation and enabling curation at the community level. *Bioinformatics*, *26*(18), 2354–2356.

Rosse, C. & Mejino, J.L.V. (2003). A reference ontology for biomedical informatics: the foundational model of anatomy. *Journal of Biomedical Informatics*, *36*(6), 478–500.

Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A. & Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, *5*(2), 51–53.

Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L.J. et al. (2007). The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, *25*(11), 1251–1255.

Smith, M.K., Welty, C. & McGuinness, D.L. (2004). OWL Web ontology language guide. Available at: http://www.w3.org/TR/owl-guide/.

Squires, B., Macken, C., Garcia-Sastre, A., Godbole, S., Noronha, J., Hunt, V., Chang, R. et al. (2008). BioHealthBase: informatics support in the elucidation of influenza virus host pathogen interactions and virulence. *Nucleic Acids Research*, *36*(Database issue), D497–D503.

The Bio Investigation Index (2009). BII: The Bio Investigation Index. Available at: http://www.ebi.ac.uk/bioinvindex/home.seam.
The OBI Consortium (2009). Available at: http://purl.obolibrary.org/obo/obi.
The Protégé Ontology Editor and Knowledge Acquisition System (2009). Available at: http://protege.stanford.edu/.
Tsarkov, D. & Horrocks, I. (2006). FaCT++ description logic reasoner: System description. In *Automated Reasoning* (pp. 292–297). Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 4130. Berlin: Springer.