WebProtégé: a Web-based Development Environment for OWL 2 Ontologies

Matthew Horridge, Tania Tudorache, Csongor Nyulas, and Mark Musen

Stanford Biomedical Informatics Research Group
Stanford University, California, USA

Abstract. We present the latest version of WebProtégé: a free, open-source Web-based tool for editing OWL ontologies. WebProtégé allows users to create, upload, share and collaboratively edit OWL ontologies. It contains various tools that are designed to support collaborative editing processes, including issue discussion, complete change tracking support and watches. Besides providing complete OWL 2 editing capabilities, WebProtégé also features a default simplified user interface that is targeted at OWL neophytes. This simplified interface, which we have designed using empirical techniques, offers a quick and easy way to edit commonly used OWL 2 axiom and class constructors. In this paper we describe these features and the main ideas behind the tool. WebProtégé is available for use at `http://webprotege.stanford.edu`.

1 Introduction

WebProtégé is a web-based, multi-user, collaborative editing environment for OWL ontologies. It is perhaps best thought of as a kind of "GoogleDocs" for ontologies. We use this analogy because WebProtégé assumes a very similar application model: When a user logs in they see a list of the ontologies that they own or the ontologies that other WebProtégé users have shared with them. They can control who can view, comment on and edit their ontologies, thus they can share ontologies with collaborators for editing or viewing. They can also make an ontology completely public so that it can be viewed and commented on by anyone who visits WebProtégé.

While WebProtégé has been around for some time—the first version was made available around 2008 [17]—this latest version was released in 2013 and is a complete rewrite. The major change is that WebProtégé is now underpinned by the OWL API [4], it now has complete support for editing OWL 2 ontologies [10], and it now features a simplified user interface that is designed to be used by OWL neophytes [6].

In this paper, we present the main motivation for WebProtégé, we examine features that are salient for editing OWL 2 ontologies, and we discuss our roadmap and planned features for the system.

2 WebProtégé Overview

When a user logs into WebProtégé they are presented with a list of the projects that they either own, or for which they have editing, commenting or viewing rights. A project is essentially a set of ontologies plus metadata, sharing settings and user interfaces settings. Users create their own projects, which are hosted on our servers at Stanford (`http://webprotege.stanford.edu`).[1] They either start by creating a new ontology, or they start by uploading a set of existing ontologies that they have already worked on. Having created a project, a user can then "share" it, adding the names of collaborators to the list of those who can edit it, comment on it, or view it.

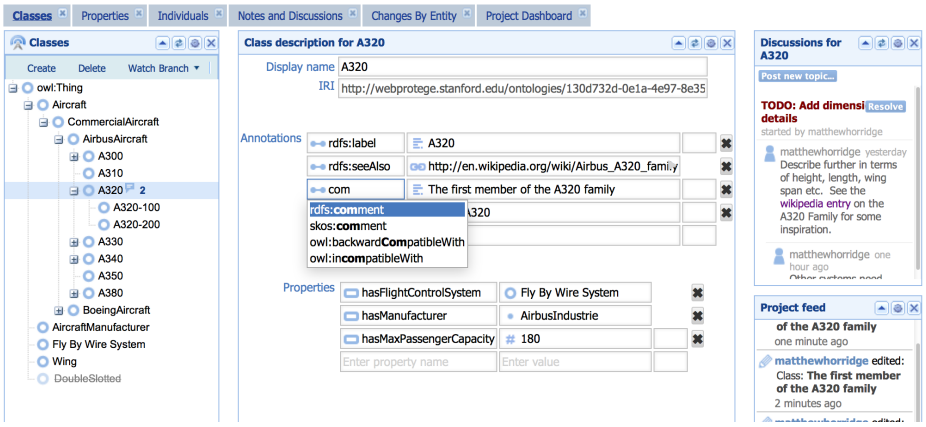

Fig. 1: The main editing interface in WebProtégé. The lefthand pane presents the class tree, indicating which classes have discussions attached to them. The middle pane presents the class frame for the selected class. The righthand pane shows the discussions for the selected class and also shows a live feed of changes.

The *default* WebProtégé user interface is shown in Figure 1. In essence it consists of a series of tabs that provide links to pre-configured *perspectives*. Each perspective consists of a layout of views known as portlets. Each *portlet* presents a specific portion of the ontologies in a project or information about a project. For example, the class hierarchy portlet displays the asserted class hierarchy (left hand side of Figure 1), the entity description portlet displays information about the selected entity (middle pane in Figure 1), while the notes and discussions portlet displays issues that pertain to the selected entity (right hand side of Figure 1).

[1] Users can also set up local WebProtégé installations if they have a desire to do so. For more details see the link to the admin guide at `https://github.com/protegeproject/webprotege/`

A key feature of the WebProtégé user interface is that project owners are able to configure their projects with custom perspectives and custom layouts within these perspectives. This means that other users see these layouts by default when they view the project in question. Furthermore, each user is able to override a project configuration in order to satisfy their own tastes and to meet the needs of the tasks that they personally have to accomplish. One benefit to this approach is that different configurations can be used to suit different groups of users that may have different levels of expertise in OWL.

3 Revision-based Change Tracking

Structured change tracking support, which records the changes made to ontologies in a project, is built into the core of WebProtégé. Changes are recorded as axiom additions and removals using structures borrowed from the OWL API. Metadata about each axiom change set is also recorded, in particular, the author, the timestamp of the change, and a high level comment that describes the change. High level comments are either automatically generated by the user interface, and are thus related to a user interface operation for example "Added A as a subclass of B", or they may be manually specified in the form of commit comments.

The change history can be viewed in a number of ways including viewing changes over a particular time period, viewing changes made by a particular author, or viewing changes that affect a particular entity (class, property, individual etc.). Figure 2 shows an example of the "changes by entity" portlet, which groups together changes that are syntactically related to the selected entity.

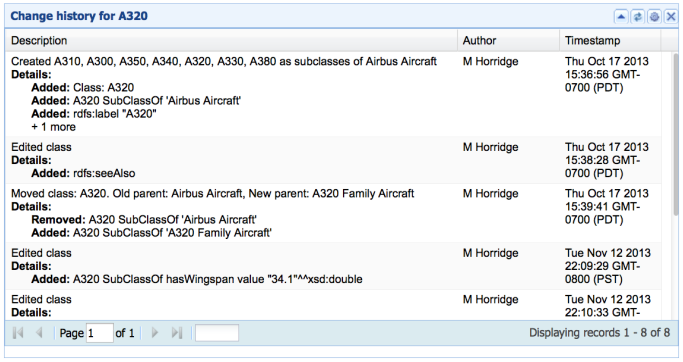

Fig. 2: The "changes by entity" portlet displaying syntactically related changes for the selected entity (A320).

The change tracking support also facilitates revision based history control. For a given project, it is possible to retrieve the ontology changes that are as-

sociated with a specific revision of that project. Although WebProtégé does not currently feature the ability to produce a diff between two versions of an ontology, the revision based change support allows prior versions of ontologies to be downloaded and compared in external diff tools such as Ecco [2], Bubbastis [9] or the OWL difference engine [13].

Finally, in relation to change tracking support, users can "watch" ontologies for changes. They can watch changes that affect the syntactic frame of a specific entity, changes that affect the frames of subclasses of a given class, or watch all possible changes to an ontology. When changes occur that fall into a scope of a watch, the user for that watch is notified via email.

4 A Simplified User Interface for OWL Neophytes

Although WebProtégé offers full-blown editing support for OWL 2 ontologies the default user interface configuration, which a user is presented with when they create a project, is the simple user interface that is shown in Figure 1. The display shown in Figure 1 is for editing class descriptions, however similar displays exists for property and individual descriptions.

This simple interface shows forms that are capable of editing a subset of the types of axiom and class constructors that are available in OWL 2. The initial design of this user interface was based on an empirical analysis of commonly used OWL constructs in a corpus of biomedical ontologies. Over the course of five months we analysed the projects that were created by users uploading existing OWL ontologies into WebProtégé in order to analyse the "fit" of the simple user interface against a wider corpus of (non-biomedical) ontologies. We found that this UI design captures much of what users need to say while shielding them from the direct use of class expressions, quantifiers and Manchester syntax details. A complete description of the design and evaluation of the UI may be found in our ISWC 2013 paper [6]; we simply present the main ideas here.

The simplified user interface that we have developed is shown as the centre pane in Figure 1 and an enlarged view is shown in Figure 3. The particular UI shown here is the main editing form for class frames. The form is composed of fields which constitute tables of property–value pairs. The fields feature auto-completion for property, class, individuals and datatype names. The auto-completion is type sensitive: it will offer only the types of entities that can be entered based on the information in the ontology up to this point. For example, the auto-completion prevents the user from entering datatypes as fillers for object property restrictions. In terms of OWL, one row in the table corresponds to one or more axioms. In the example in Figure 3, the row hasFlightControlSystem and FlyByWireSystem corresponds to the axiom SubClassOf(:A320, ObjectSomeValuesFrom(:hasFlightControlSystem :FlyByWireSystem)).

A key feature of this simplified UI is that it minimises the distinctions that users have to make explicitly. For example, in previous versions of the tool [16], when a user created a new property, they had to decide explicitly whether the property was an object or a data one. Similarly, when entering class expressions

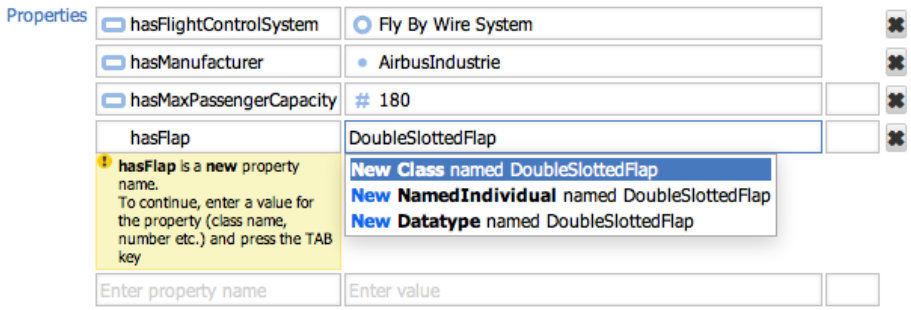

Fig. 3: Property–value pairs being edited. The class frame in the figure contains mixed object and data property usage. It also contains a mix of ObjectSomeValuesFrom, ObjectHasValue and DataHasValue class expressions. The auto-completion box prompts the user to create new entities where necessary. We eliminated the need to choose explicitly between object and data properties; we determine property types based on filler values.

the user had to make various choices such as choosing between SomeValuesFrom and AllValuesFrom restrictions, and between SomeValuesFrom and HasValue restrictions. In this simplified UI, we use simple and reliable heuristics to determine the type of property and the type of restriction that the user creates based on the fillers that she specifies. Figure 3 displays a class description that has mixed use of data and object properties. It also contains mixed use of different types of class expressions, individuals, and data values: the first row corresponds to an ObjectSomeValuesFrom class expression whose filler is a class, the second row an ObjectHasValue class expression whose filler is an individual, and the third row a DataHasValue class expression whose value is an integer literal. At no point when entering the information shown in Figure 3 has the user *explicitly* had to decide upon and choose the types of class expressions, or decide upon and choose the types of properties—the system determines these distinctions in a straightforward but highly effective way.

Finally, this UI also supports a kind of on-the-fly object creation and type inference. In the fourth row in Figure 3 the user wants to specify a new type of flap for the class (aircraft) they are describing. However, hasFlap is a new property name. In this case, the system accepts the new property name, warns the user that it is new (in case the user has simply made a typo) and allows them to move on to specify a filler. In this case, the user specifies a new class (DoubleSlottedFlap). Once they enter this information, WebProtégé creates the necessary declarations of the appropriate type and generates the class expressions and axioms under the hood.

In addition to editing logical information, WebProtégé provides support for describing extra-logical information about entities through OWL annotations. These annotations are part of the class frame (Figure 1). WebProtégé provides

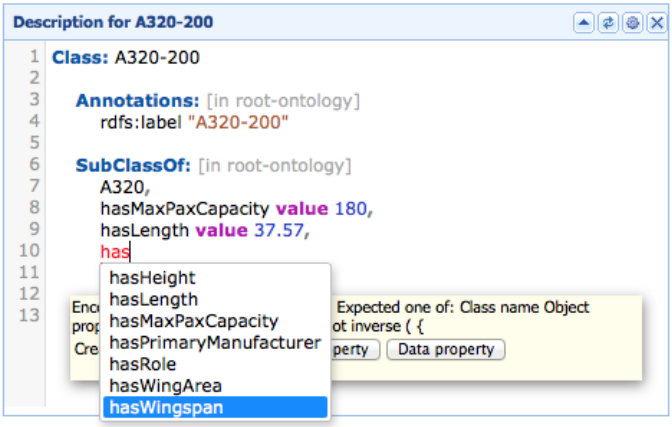

Fig. 4: Editing an OWL 2 class description using auto-completion to complete class, property and individual names.

auto-completion support that allows users to reuse annotation vocabulary from well known metadata sets such as DublinCore and SKOS [1] (Figure 1).

5 Complete Editing Support for OWL 2 Ontologies

At the other end of the scale from the simplified UI, WebProtégé provides complete syntactic editing support for OWL 2 ontologies. One of the main changes in this latest version of WebProtégé is that it underpinned by the OWL API—the de-facto standard API for working with OWL 2 ontologies. This means that WebProtégé "natively" supports OWL 2 both on the back-end and on the front-end. Indeed, the user interface can be configured with components that allow OWL 2 entity frames to be edited. Descriptions are presented using a slightly extended version of the Manchester OWL Syntax [5]. The extensions, which are minor, allow source ontologies in an imports closure to be specified for sets of axioms. For example, in Figure 4, the syntax "[in root-ontology]" specifies that associated annotation assertions and subclass axioms reside in the project root ontology (the root of the project imports closure). This small extension allows axioms to easily be moved between ontologies in an imports closure.

Figures 4, 5 and 6 show different aspects of the main view for full-blown OWL 2 entity description editing. As can be seen, WebProtégé supports integrated development environment (IDE) style code editing. It offers many of the standard code features that are expected of IDEs.

Auto-completion As a user types in expressions they can invoke the auto-completion facility (Figure 4) so that they can easily re-use existing entity names and built in keywords. Auto-completion is context sensitive so that it only of-

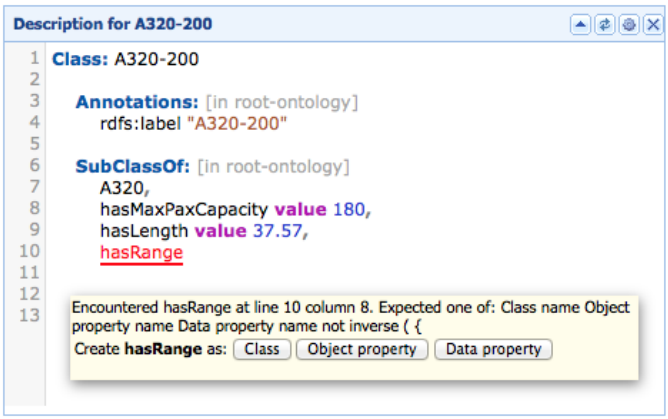

Fig. 5: Error highlighting whilst editing an OWL 2 class description. WebProtégé features syntax checking and error highlighting for class descriptions written in the Manchester OWL Syntax. If a name is not recognised, the user interface prompts the user to create the name as a new type of class, object property, data property, annotation property, datatype, or individual as appropriate.

fers names corresponding to entity types that can be entered into the current location.

Error checking and on-the-fly object creation Descriptions of entities are checked as a user enters them into the editing area. If unrecognised entity names are encountered they are highlighted in red (Figure 5). Unlike the desktop version, WebProtégé allows inline entity creation. For example, in Figure 5, the user has typed hasRange, but this name is not recognised as it is not bound to an existing entity. At this point the user can decide that hasRange is indeed a new entity and they can choose to create it on the fly—in this case as a data-property.

Change committing Once a user has finished editing the description of an entity they can choose to "commit" the changes to the project so that other users can see them. As shown in Figure 6, they can either choose to commit the changes with or without a high level comment. This allows users to elaborate on the reasons for multiple, possibly complex changes in the context of a change to an entity description. If a user chooses not to explicitly commit changes they will be auto-committed when the selected entity changes.

6 Form-based Ontology Editing

Another major feature of the UI, which has successfully been used in a very large project that is producing the next version of the International Classification of Diseases [14, 15], is that it can display fine-grained customised Web-forms for

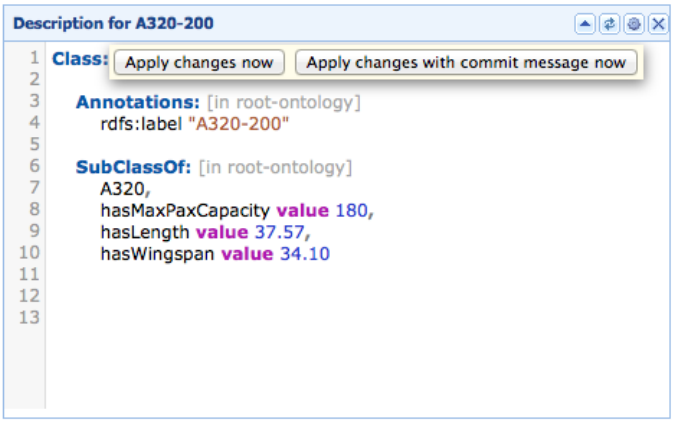

Fig. 6: When a user has finished editing a class description they get prompted to commit the changes that they have made. The user can commit the changes with a default commit message, or they can specify a custom message that can be used to describe the changes and the rationale for them in more detail.

editing descriptions of entities and for instance acquisition. An example of such a form is shown in Figure 7. The form is created with a markup language that allows the layout of widgets to be specified along with a description of how these widgets should behave with respect to the underlying ontology.

In contrast to the forms in Protégé 3, which are automatically generated from the TBox in an ontology, the forms in WebProtégé are designed by hand and manually linked to the underlying ontology. While this requires some initial setup, we have found that there are several benefits to this loosely coupled approach. In particular, the underlying ontology does not need to be polluted with a tangled mess of expressive axioms or baroque modelling choices in order to get "the forms to come out right". For example, it is not necessary to make a property functional in order to make the form display radio buttons instead of check boxes, or a drop down box instead of a list box. Similarly, domains need not be specified in order to get entry fields for properties on a form, and ranges need not be specified to restrict values for a form field to specific entities in an ontology.

7 Current and Future Work

Reasoning support Our top priority at the moment is OWL reasoner integration. We intend to offer cloud-based reasoning to support to projects that require it (resource permitting). The end functionality will be similar to the functionality that is offered in the desktop suite. That is, WebProtégé will allow users to check the consistency of ontologies in their project, all users to view an entailed class and property hierarchies, and entailed types for individuals. We

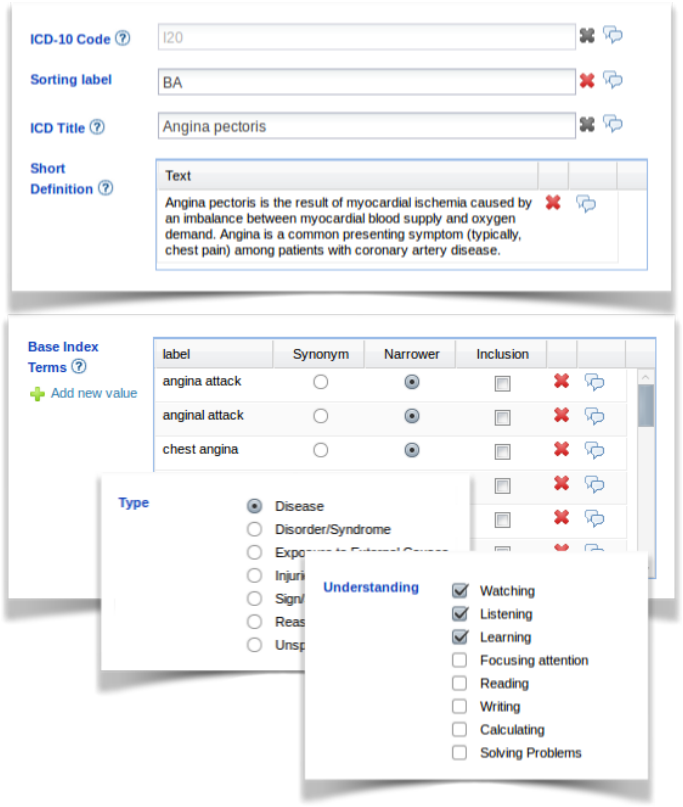

Fig. 7: A sample of some of the forms used to edit the International Classification of Diseases (ICD) ontology. The forms are created using a markup language, which specifies the form controls to use, the layout and how to bind these controls to the underlying ontology. This lose coupling offers a great deal of flexibility.

will also support DL queries in a similar manner to the DL query tab in the desktop version.

We envisage that, at least by default, users will not interact with a reasoner directly. Instead, a reasoner will be enabled for a project and it will run in the background, with consistency checking and satisfiability checking being performed automatically against each project revision. As users perform edits the reasoner will automatically catch up and users will be able to query against the latest revision that has been processed by the reasoner.

Integration with third party tools such as GitHub In the software engineering world the software "Git" is rapidly becoming the tool of choice for

version control and collaborative software engineering projects. GitHub[2] is a social coding website that hosts Git-based projects and provides tools such as wikis, release pages and issue trackers. While GitHub primarily hosts software projects, it is also used by some ontology authors in the Biomedical Ontology Community for hosting ontology-based projects. For example, the Uber Anatomy Ontology (UBERON) project [3] is hosted on GitHub.[3] Projects such as UBERON take advantage of the GitHub's social coding tools, in particular issues trackers, and also use it to keep track of previous versions. Given the rising popularity of GitHub-based ontology projects, it is our intention to provide some kind of integration with tools like GitHub. In particular, we imagine that project releases could be pushed from WebProtégé to an associated GitHub repository and that GitHub issue trackers could also be linked to from within WebProtégé.

Support for different ontological sources Recent work on tools such as RightField [18], Populous [7] and Tawny OWL [8] has taken a somewhat different approach to ontology editing for domain experts than traditional tools such as Protégé. Rather than having domain experts directly edit and manipulate ontologies in ontology editors, these tools have them edit raw source material for ontologies, which then gets "compiled" into OWL. This raw source material has take the form of spreadsheets and comma separated value (CSV) files in tools such as RightField and Populous, and in the case of Tawny-OWL it has taken the form of a particular dialect of the Clojure programming language. We believe that this kind of approach has certain advantages over direct ontology manipulation. In particular, domain experts can work with tools such as MS-Excel that they have experience with and are comfortable working with; they can perform bulk manipulation or bulk data entry if working with spreadsheet like tools; and finally, the translation to OWL can be specified in a standoff way that can be re-run, altered, optimised and experimented with in order to produce an end ontology that meets application requirements. While WebProtégé currently supports a form of bulk data entry whereby users can work in Excel and import the resulting spreadsheets and CSV files into the system, we intend to evolve this idea so that multiple different source file formats can be used. We can also imagine that data entered through WebProtégé forms could be stored in its raw-data state, which could later be compiled into OWL, thus allowing a flexible translation process. Ultimately, we envisage that any translation and import process will be declaratively specified using a tool such as Mapping Master [12, 11].

8 Availability and Uptake

WebProtégé is open source and freely available at `http://webprotege.stanford.edu`. We encourage potential users to take advantage of this hosted solution as it can be used with zero setup costs. Furthermore, we have a strict privacy policy;

[2] `http://www.github.com`
[3] https://github.com/obophenotype/uberon

we regularly update it in a seamless fashion so that it has the latest bug fixes and features; and we provide nightly project backup.

At the time of writing the version of WebProtégé hosted at `http://webprotege.stanford.edu` contains just over 10,000 ontology projects. While there are many small ontologies, the largest ontologies that have been loaded into the system are hundreds of thousands of axioms in size and there are projects that have tens of thousands of changes.

For users who are unable to use our hosted solution, WebProtégé is also freely available for download from our GitHub site: `http://www.github.com/protegeproject/webprotege`. It can be set up on a local network, which is convenient for users working behind a corporate firewall, for example.

Finally, WebProtégé is written using the Google Web Toolkit. All code and developer documentation is available on our GitHub site.

Acknowledgements This work was supported by Grants GM103316 and R01GM086587 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

References

1. Thomas Baker, Sean Bechhofer, Antoine Isaac, Alistair Miles, Guus Schreiber, and Ed Summers. Key choices in the design of simple knowledge organization system (SKOS). *Journal of Web Semantics*, 20:35–49, 2013.
2. Rafael S. Gonçalves, Bijan Parsia, and Ulrike Sattler. Ecco: A hybrid diff tool for OWL 2 ontologies. In Pavel Klinov and Matthew Horridge, editors, *Proceedings of OWL: Experiences and Directions Workshop 2012*, volume 849 of *CEUR Workshop Proceedings*, 2012.
3. Melissa Haendel, James P. Balhoff, Frederic B. Bastian, David C. Blackburn, Judith A. Blake, Yvonne Bradford, Aurélie Comte, Wasila M. Dahdul, Thomas Dececchi, Robert E. Druzinsky, Terry F. Hayamizu, Nizar Ibrahim, Suzanna E. Lewis, Paula M. Mabee, Anne Niknejad, Marc Robinson-Rechavi, Paul C. Sereno, and Christopher J. Mungall. Unification of multi-species vertebrate anatomy ontologies for comparative biology in uberon. *Journal of Biomedical Semantics*, 5:21, 2014.
4. Matthew Horridge and Sean Bechhofer. The OWL API: A Java API for OWL ontologies. *Semantic Web*, 2(1):11–21, February 2011.
5. Matthew Horridge, Nick Drummond, John Goodwin, Alan L. Rector, Robert Stevens, and Hai Wang. The Manchester OWL syntax. In Bernardo Cuenca Grau, Pascal Hitzler, Conor Shankey, and Evan Wallace, editors, *OWLED*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.
6. Matthew Horridge, Tania Tudorache, Jennifer Vendetti, Csongor Nyulas, Mark A. Musen, and Natalya Fridman Noy. Simplified OWL ontology editing for the Web: Is WebProtégé enough? In Harith Alani et al, editor, *International Semantic Web Conference (1)*, volume 8218 of *Lecture Notes in Computer Science*, pages 200–215. Springer, 2013.
7. Simon Jupp, Matthew Horridge, Luigi Iannone, Julie Klein, Stuart Owen, Joost Schanstra, Katy Wolstencroft, and Robert Stevens. Populous: a tool for building OWL ontologies from templates. *BMC Bioinformatics*, 13(S-1):S5, 2012.

8. Phillip Lord. The semantic web takes wing: Programming ontologies with Tawny-OWL. In Mariano Rodriguez-Muro, Simon Jupp, and Kavitha Srinivas, editors, *Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013) co-located with 10th Extended Semantic Web Conference (ESWC 2013), Montpellier, France, May 26-27, 2013*, volume 1080. CEUR-WS.org, 2013.
9. James Malone, Ele Holloway, Tomasz Adamusiak, Misha Kapushesky, Jie Zheng, Nikolay Kolesnikov, Anna Zhukova, Alvis Brazma, and Helen Parkinson. Modelling sample variables with and experimental factor ontology. *Bioinformatics*, 26(8):1112–1118, March 2010.
10. Boris Motik, Peter F. Patel-Schneider, and Bijan Parsia. OWL 2 Web Ontology Language structural specification and functional style syntax. W3C Recommendation, W3C – World Wide Web Consortium, October 2009.
11. Martin J. O'Connor, Christian Halaschek-Wiener, and Mark A. Musen. M^2: A language for mapping spreadsheets to OWL. In Evren Sirin and Kendall Clark, editors, *OWLED*, volume 614 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.
12. Martin J. O'Connor, Christian Halaschek-Wiener, and Mark A. Musen. Mapping master: A flexible approach for mapping spreadsheets to OWL. In Peter F. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, Jeff Z. Pan, Ian Horrocks, and Birte Glimm, editors, *International Semantic Web Conference (2)*, volume 6497 of *Lecture Notes in Computer Science*, pages 194–208. Springer, 2010.
13. Timothy Redmond and Natalya F. Noy. Computing the changes between ontologies. In Vit Novacek, Zhisheng Huang, and Tudor Groza, editors, *EvoDyn 2011 Knowledge Evolution and Ontology Dynamics.*, volume 784 of *CEUR Workshop Proceedings*. CEUR-WS.org, October 2011.
14. Tania Tudorache, Sean M. Falconer, Natalya Fridman Noy, Csongor Nyulas, Tevfik Bedirhan Üstün, Margaret-Anne D. Storey, and Mark A. Musen. Ontology development for the masses: Creating ICD-11 in webprotégé. In Philipp Cimiano and Helena Sofia Pinto, editors, *EKAW*, volume 6317 of *Lecture Notes in Computer Science*, pages 74–89. Springer, 2010.
15. Tania Tudorache, Csongor Nyulas, Natalya Fridman Noy, and Mark A. Musen. Using semantic web in ICD-11: Three years down the road. In Harith Alani et al, editor, *International Semantic Web Conference (2)*, volume 8219 of *Lecture Notes in Computer Science*, pages 195–211. Springer, 2013.
16. Tania Tudorache, Csongor Nyulas, Natalya Fridman Noy, and Mark A. Musen. WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web. *Semantic Web*, 4(1):89–99, 2013.
17. Tania Tudorache, Jennifer Vendetti, and Natalya Fridman Noy. WebProtege: A lightweight owl ontology editor for the web. In Catherine Dolbear, Alan Ruttenberg, and Ulrike Sattler, editors, *OWLED*, volume 432 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
18. Katy Wolstencroft, Stuart Owen, Matthew Horridge, Wolfgang Müller, Finn Bacall, Jacky L. Snoep, Franco du Preez, Quyen Nguyen, Olga Krebs, and Carole A. Goble. Rightfield: Scientific knowledge acquisition by stealth through ontology-enabled spreadsheets. In Annette ten Teije et al, editor, *Knowledge Engineering and Knowledge Management - 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings*, volume 7603 of *Lecture Notes In Computer Science*, pages 438–441. Springer, 2012.