Simplified OWL Ontology Editing for the Web: Is WebProtégé Enough?

Matthew Horridge, Tania Tudorache, Jennifer Vendetti, Csongor Nyulas, Mark A. Musen, and Natalya F. Noy

Stanford Center for Biomedical Informatics Research
Stanford University, Stanford, CA, 94305, USA
{horridge,tudorache,vendetti,nyulas,musen,noy}@stanford.edu

Abstract. Ontology engineering is a task that is notorious for its difficulty. As the group that developed Protégé, the most widely used ontology editor, we are keenly aware of how difficult the users perceive this task to be. In this paper, we present the new version of WebProtégé that we designed with two main goals in mind: (1) create a tool that will be easy to use while still accounting for commonly used OWL constructs; (2) support collaboration and social interaction around distributed ontology editing as part of the core tool design. We designed this new version of the WebProtégé user interface empirically, by analysing the use of OWL constructs in a large corpus of publicly available ontologies. Since the beta release of this new WebProtégé interface in January 2013, our users from around the world have created and uploaded 519 ontologies on our server. In this paper, we describe the key features of the new tool and our empirical design approach. We evaluate language coverage in WebProtégé by assessing how well it covers the OWL constructs that are present in ontologies that users have uploaded to WebProtégé. We evaluate the usability of WebProtégé through a usability survey. Our analysis validates our empirical design, suggests additional language constructors to explore, and demonstrates that an easy-to-use web-based tool that covers most of the frequently used OWL constructs is sufficient for many users to start editing their ontologies.

1 Introduction

"Protégé is too difficult to use!" The Protégé team hears this sentiment from our users all too often. As we observe many of them grapple with the difficulties of learning a highly expressive logic-based ontology language such as OWL, we see how onerous ontology development can be. Other studies on cognitive complexity of ontology development bear out these observations [1].

Developers of tools for ontology browsing and editing have faced the dilemma: On the one hand, we want to support international standards, such as OWL 2, fully in order to ensure interoperability [2]. On the other hand, we want to make sure that both beginners and experts alike can develop ontologies easily.

In this paper, we report on our design and evaluation of a major new release of WebProtégé, a web-based version of Protégé that uses the OWL API [3] and that provides support for editing OWL 2 ontologies. We released this new version of WebProtégé in January 2013, and we had two main goals for the design of this version: (1) create a tool that will be easy to use while still accounting for commonly used OWL constructs; (2) support distributed ontology editing, collaboration, and interaction as part of the core tool design. The new WebProtégé serves as a "Google docs" environment for ontologies, enabling users to upload their ontologies, to initiate new projects, and to invite their collaborators to participate in the development. We have created a cut-down user interface in WebProtégé, which makes creating new classes or updating information as simple as filling out a web form. This interface is the default interface that WebProtégé users see when they create or upload an OWL ontology. The users have the option of enabling more advanced features. In this paper, we discuss the design and evaluation of the choice of language constructs supported in this default interface, assuming that the default interface is what the majority of our users will see.

We address the following research questions in this paper: (1) Is there a subset of OWL that accounts for the majority of term descriptions used by ontology developers in various scientific domains? (2) How do we design a user interface that enables efficient editing of the most common constructs while providing an opportunity for the more expert users to access as much of the advanced features as possible?

In order to address these questions, we start by analysing a corpus of 330 publicly available ontologies in BioPortal [4, 5] to determine which OWL constructs ontology developers use most frequently (Section 4). We use the results of this analysis to determine which set of features to include in the default configuration of WebProtégé. We evaluate this new release in two ways. First, we evaluate the *coverage* of the constructs supported by the user interface by analysing the aggregated information about the ontologies that WebProtégé users uploaded to the WebProtégé server. This new corpus constitutes a set of ontologies that were created elsewhere and thus presents a "naturally occurring" corpus of ontologies. Second, we conducted a survey of the users of the new tool in order to evaluate the usability of the tool; to understand what the users like and do not like about the tool; and to gauge whether or not the users feel limited by the default interface or whether they feel that they can perform all of the editing tasks that they need to perform (Section 6).

This paper makes the following contributions: (1) We present an empirical methodology for developing an easy-to-use ontology editor based on analysing a large training corpus of ontologies. (2) We evaluate the *language coverage* provided the user interface in WebProtégé by analysing the OWL constructs in a test corpus of 230 ontologies that WebProtégé users have created in other tools and uploaded to WebProtégé. (3) We evaluate the *usability* of WebProtégé through a usability questionnaire that close to 20% of WebProtégé users have answered.

Many of the lessons that we learn from designing and evaluating WebProtégé are not specific to our tool. Indeed, our paper analyses the broader question of how we can use a principled approach to make ontology editing easier and whether simplified ontology editing is indeed possible, practical, and useful.

2 Preliminaries

In the work presented here, we deal with ontologies written in the Web Ontology Language (OWL), and more specifically OWL 2, its latest version [6]. Throughout the rest of this paper we refer to OWL 2 simply as OWL. In this section, we present the main OWL terminology that is useful in the context of this paper. We assume that the reader has basic familiarity with ontologies and OWL.

OWL and Ontologies An OWL ontology is a set of *axioms*. Each axiom makes a *statement* about the domain of interest. The building blocks of axioms are *entities* and *class expressions*. Entities correspond to the important terms in the domain of interest and include *classes*, *properties*, *individuals*, and *datatypes*. Properties may be subdivided into *object*, *data*, and *annotation* properties. The *signature* of an ontology is the set of entities that appear in that ontology. OWL is a highly expressive language and features a rich set of class constructors that allow entities to be combined into more *complex class expressions*. As a convention, we use the letters A and B to stand for class names and the letters C and D to stand for (possibly complex) class expressions. In this paper, we largely focus on subclass axioms SubClassOf(C, D), equivalent class axioms EquivalentClasses(C, D), disjoint classes axioms DisjointClasses(C, D) and annotation assertions AnnotationAssertion(P, A, v). We refer to SubClassOf, EquivalentClasses and DisjointClasses axioms as *logical* axioms and AnnotationAssertion axioms as *non-logical* axioms. We also focus on two broad types of class expressions: (1) class expressions that we loosely term *existential restrictions*, which specify the existence of relationships between individuals and by which we mean ObjectSomeValuesFrom(R, C), DataSomeValuesFrom(R, C), ObjectHasValue(R, a), and DataHasValue(R, l) restrictions; and (2) class expressions that we term *universal restrictions*, by which we mean ObjectAllValuesFrom(R, C) and DataAllValuesFrom(R, C) restrictions.

Frame-Based Views of Ontologies Even though an OWL ontology is simply a set of axioms, few ontology-development environments choose to display ontologies as lists of axioms. Most environments are *entity-centric* and revolve around the idea of editing *descriptions* of entities. In essence, when an entity is selected in a tool like Protégé, the tool presents (a partial view of) the subset of axioms that describe or define that entity. We call such a subset of axioms an *entity-frame*, or more specifically a *class-frame* for a class and so on. In this work we focus on class-frames, which we define as follows:

Definition 1 (Class-Frame). *For a class A in the signature of an ontology $\mathcal{O}$ the class-frame for A w.r.t. $\mathcal{O}$ is the subset-maximal set of axioms $\mathcal{S} \subseteq \mathcal{O}$ where each axiom in $\mathcal{S}$ is of the form SubClassOf(A, C), EquivalentClasses(A, C), DisjointClasses(A, C) and AnnotationAssertion(P, A v), where C is a (possibly complex) class expression, P is an annotation property, and v is an annotation value (literal, IRI or anonymous individual).*

OBO and OWL In the world of biomedical ontologies, there is another, widely used language, called OBO [7]. There is a close relationship between OBO and OWL 2, and it is possible to translate faithfully the logical aspects of an OBO ontology into an OWL 2 ontology [8]. For the purposes of the work here we therefore view OBO as a syntactic variant of OWL 2.

3 An Overview of WebProtégé

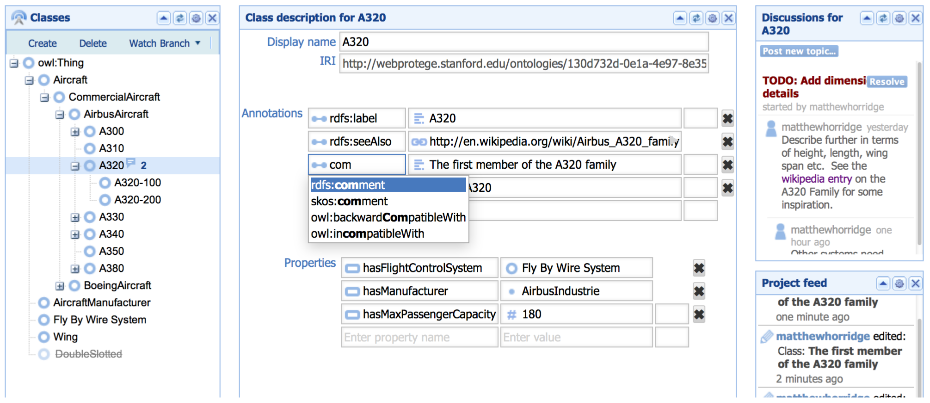
This section presents a high level overview of WebProtégé and its salient features. The main purpose of this section is to provide a context for the discussion on our empirically driven user interface (UI) design in Section 4.

WebProtégé is a web-based, multi-user, collaborative editor for OWL ontologies. The main document unit in WebProtégé is a *project*, which is a set of ontologies plus metadata, sharing settings and UI settings. Users create their own projects, which are hosted on our servers at Stanford.[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 then "shares" this project, adding the names of her collaborators to the list of those who can edit her ontology. Now, any time the user or any of her collaborators logs into WebProtégé, she can see her ontology under development. As one of the users creates or edits the ontology, others can see the changes immediately. They can comment on the changes and carry out discussions in the tool—with the discussions linked to the class that they are discussing. If they log in after a few days, they can see the summary of changes to the ontologies and to the classes on their "watch list." As the project matures, they can invite others to participate and to comment, or choose to publish the ontology in a public repository for the broader community to use. They can download any revision of their ontology and process it using any other OWL tools such as reasoners, visualisation, and query tools.

Figure 1 shows a screenshot of the main editing interface in WebProtégé. The left pane consists of a tree for navigating the class hierarchy and for selecting a class frame for editing. The middle pane captures a subset of the selected class frame. We provide a precise description of and the rationale for what this frame captures in Section 4. The right pane in Figure 1 contains tools for collaboration. In particular, it shows a threaded list of issues and discussions and a live activity feed. Users can configure all elements in the interface, augment it with different views or reconfigure it completely to suit their needs.

The centre pane in Figure 1 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

[1] Users can also set up local WebProtégé installations if they have a desire to do so.


Classes

Create Delete Watch Branch

- owl:Thing
  - Aircraft
    - CommercialAircraft
      - AirbusAircraft
        - A300
        - A310
        - A320 2
          - A320-100
          - A320-200
        - A330
        - A340
        - A350
        - A380
      - BoeingAircraft
  - AircraftManufacturer
  - Fly By Wire System
  - Wing
  - DoubleSlotted

Class description for A320

Display name A320

IRI http://webprotege.stanford.edu/ontologies/130d732d-0e1a-4e97-8e35

Annotations

rdfs:label	A320
rdfs:seeAlso	http://en.wikipedia.org/wiki/Airbus_A320_family
com	The first member of the A320 family
	A320

rdfs:comment
skos:comment
owl:backwardCompatibleWith
owl:incompatibleWith

Properties

hasFlightControlSystem	Fly By Wire System
hasManufacturer	AirbusIndustrie
hasMaxPassengerCapacity	180
Enter property name	Enter value

Discussions for A320

Post new topic...

TODO: Add dimensi Resolve
details
started by matthewhorridge

matthewhorridge yesterday
Describe further in terms of height, length, wing span etc. See the wikipedia entry on the A320 Family for some inspiration.

matthewhorridge one hour ago

Project feed

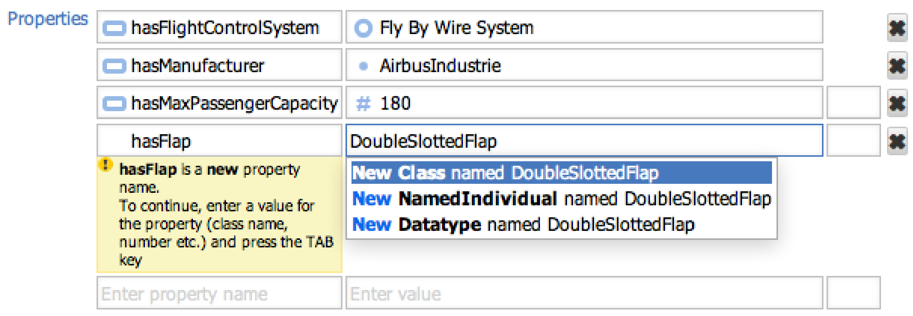
of the A320 family
one minute ago

matthewhorridge edited:
Class: The first member of the A320 family
2 minutes ago

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. The righthand pane shows the discussions for the class and the live feed of changes.

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 2, the row hasFlightControlSystem and FlyByWireSystem corresponds to the axiom SubClassOf(:A320, ObjectSomeValuesFrom(:hasFlightControlSystem :FlyByWireSystem)).

A key feature of the WebProtégé UI is that it minimises the distinctions that users have to make explicitly. For example, in previous versions of the tool [9], when a user created a new property, she had to decide explicitly whether the property was an object or a data one. Similarly, when entering class expressions the user had to make various choices such as choosing between SomeValuesFrom and AllValuesFrom restrictions, and between SomeValuesFrom and HasValue restrictions. In the WebProtégé 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 2 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 2 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 2 the user wants to specify a new type of flap for the class (aircraft) that she is 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 her to move on to specify a filler. In this case, she specifies a



Properties

hasFlightControlSystem	Fly By Wire System	
hasManufacturer	AirbusIndustrie	
hasMaxPassengerCapacity	180	
hasFlap	DoubleSlottedFlap	

hasFlap is a **new** property name.
To continue, enter a value for the property (class name, number etc.) and press the TAB key

New Class named DoubleSlottedFlap
New NamedIndividual named DoubleSlottedFlap
New Datatype named DoubleSlottedFlap

Enter property name
Enter value

Fig. 2. 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.

new class (DoubleSlottedFlap). Once the user enters 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 (Figures 1 and 2). WebProtégé provides auto-completion support that allows users to reuse annotation vocabulary from well known metadata sets such as DublinCore and SKOS (Figure 1).

4 The WebProtégé Profile (WPP)

The following are our main high level design goals for WebProtégé: (1) to provide an ontology-editing infrastructure with zero installation and zero setup costs, (2) to provide a framework that supports multiple editors, commenters and viewers to work simultaneously on the same ontology; (3) to provide a simple UI that allows novices and experts alike to enter information in a way that is comfortable for them. Developing WebProtégé as a Web-app achieves the first goal and goes some of the way in supporting collaboration. In this section, we look at the simple UI that WebProtégé provides, what exactly it can represent and, how we arrived at this current design. We call the set of language features supported by the UI the *WebProtégé Profile* (WPP).

Definition of the WebProtégé Profile

Although WebProtégé supports the editing of class, property, and individual frames, we focus our discussion on class frames. We focus on class frames because property frames are somewhat simpler than class frames, with fewer design choices to make, and individual frames are themselves similar to class frames.

The default class frame editor in WebProtégé supports editing class frames defined as follows.

Definition 2 (WPP). *A WebProtégé Profile class frame for a class A in the signature of an ontology $\mathcal{O}$ is the subset-maximal set of axioms $\mathcal{S} \subseteq \mathcal{O}$ such that each axiom in $\mathcal{S}$ conforms to the following grammar, where non-terminals are shown in bold, terminals are shown in a regular font-weight surrounded by single quotes, choices are indicated with a bar, zero or more items are shown in curly brackets. The non-terminals* ***Class****,* ***ObjectProperty****,* ***DataProperty****,* ***AnnotationProperty****,* ***NamedIndividual****,* ***Datatype****,* ***Literal*** *and* ***IRI****, are defined as they appear in the OWL 2 Structural Specification.*

$$
\begin{array}{rl}
\textbf{ClassFrame} := & \{\textbf{ClassFrameAxiom}\} \\
\textbf{ClassFrameAxiom} := & \text{`SubClassOf'}\ \text{`('}\ A\ \textbf{ClassExpression}\ \text{`)'}\ | \\
 & \text{`AnnotationAssertion'}\ \text{`('}\ \textbf{AnnotationProperty}\ A\ \textbf{AnnoValue}\ \text{`)'} \\
\textbf{ClassExpression} := & \textbf{Class}\ | \\
 & \text{`ObjectIntersectionOf'}\ \text{`('}\ \textbf{ClassExpression}\ \textbf{ClassExpression}\ \{\textbf{ClassExpression}\}\ \text{`)'}\ | \\
 & \text{`ObjectSomeValuesFrom'}\ \text{`('}\ \textbf{ObjectProperty},\ \textbf{Class}\ \text{`)'}\ | \\
 & \text{`ObjectSomeValuesFrom'}\ \text{`('}\ \textbf{ObjectProperty},\ \text{`\{'}\ \textbf{NamedIndividual}\ \text{`\}'}\ \text{`)'}\ | \\
 & \text{`ObjectHasValue'}\ \text{`('}\ \textbf{ObjectProperty},\ \textbf{NamedIndividual}\ \text{`)'}\ | \\
 & \text{`DataSomeValuesFrom'}\ \text{`('}\ \textbf{DataProperty},\ \textbf{Datatype}\ \text{`)'}\ | \\
 & \text{`DataSomeValuesFrom'}\ \text{`('}\ \textbf{DataProperty},\ \text{`\{'}\ \textbf{Literal}\ \text{`\}'}\ \text{`)'}\ | \\
 & \text{`DataHasValue'}\ \text{`('}\ \textbf{DataProperty},\ \textbf{Literal}\ \text{`)'}\ | \\
 & \text{`ObjectMinCardinality'}\ \text{`('}\ \text{`1'}\ \textbf{ObjectProperty},\ \textbf{Class}\ \text{`)'}\ | \\
 & \text{`DataMinCardinality'}\ \text{`('}\ \text{`1'}\ \textbf{DataProperty},\ \textbf{Class}\ \text{`)'} \\
\textbf{AnnoValue} := & \textbf{Literal}\ |\ \textbf{IRI}
\end{array}
$$

Definition 2 (WPP) precisely represents the language that is supported by the default class frame editor in WebProtégé. We chose what to include in the Definition 2 (WPP) based on (1) an empirical analysis of commonly used axiom types and class constructors in a large ontology corpus, and (2) commonly reported errors [10, 11] that are made by novices when building OWL ontologies. The corpus analysis provided information on which constructs we should support. The error analysis helped us to decide which decisions we should take out of the hands of novice users.

An Analysis of Constructs from the BioPortal Ontology Corpus

BioPortal is a community-based repository of biomedical ontologies [4].[2] At the time of writing, it contains more than 330 public ontologies with almost six million terms in them. We used OWL and OBO ontologies from BioPortal as our corpus to analyse the commonly used OWL constructs.

While BioPortal contains only the ontologies that are developed by researchers and practitioners in biomedicine, it is still an attractive corpus for a general-purpose analysis for the following reasons: First, ontologies in BioPortal vary

[2] `http://bioportal.bioontology.org`

Table 1. Class frame axioms and class constructor occurrences in the BioPortal corpus. The corpus contains 261 ontologies in OWL and OBO. Pn represents the n^{th} percentile number of occurrences of a particular construct. For example, a P25 of 185 for SubClassOf axioms means that 25% of ontologies contain 185 SubClassOf axioms or less. The category Existential includes ObjectSomeValuesFrom, DataSomeValuesFrom, ObjectHasValue, and DataHasValue class expressions. The category Universal includes ObjectAllValuesFrom and DataAllValuesFrom class expressions. MinCardinality, MaxCardinality and ExactCardinality combine both object and data cardinality restrictions.

Constructor Type	**# of**	**% of**	**# occurrences of constructors**				
	ontologies	**ontologies**	**P25**	**P50**	**P75**	**P90**	**Max #**
SubClassOf	243	93.1	185	521	2705	12,309	847,755
EquivalentTo	80	30.7	4	16	61	403	73,461
DisjointWith	82	31.4	3	28	158	673	56,192
Existential	162	62.1	37	157	1,461	9,651	641,123
Universal	45	17.2	4	22	49	145	22,371
Object Union	64	24.5	3	7	20	65	387
Object Complement Of	19	7.3	1	4	15	35	99
Object One Of	8	3.1	1	1	4	4	5
MinCardinality	28	10.7	1	3	5	14	1,305
MaxCardinality	10	3.8	1	3	10	110	967
ExactCardinality	23	8.8	4	10	20	23	257

greatly in size and expressivity [12]. Second, these ontologies are naturally occurring ontologies, and they are developed by a wide range of groups and ontology engineers. Finally, biomedical ontologies account for a large fraction of ontologies under development in tools such as Protégé. Therefore, it seems reasonable that, if we can provide a UI that accommodates a large proportion of the BioPortal ontologies, then that UI will also satisfy a large number of potential WebProtégé users.

Materials and Method We accessed BioPortal on August 31, 2012 using the NCBO Web services API [5]. We downloaded all OWL compatible (OWL plus OBO) ontologies. There were 261 such ontologies. We used the OWL API (version 3.4.0) to parse and analyse each ontology. We recorded the number and kinds of class frame axioms (SubClassOf, EquivalentClasses, DisjointWith) for each ontology, as well as the number of occurrences of the different kinds of OWL class expressions.

Results Table 1 shows the occurrences of class frame axioms and class expressions. For each type of constructor, the table presents the number and percentage of ontologies that contain that constructor and the 25th, 50th, 75th, 90th percentile values (over the ontologies containing that constructor), and maximum occurrences per ontology.

Analysis It is clear from Table 1 that SubClassOf is the dominant form of axiom type. Most ontologies (93%) contain these types of axioms. By contrast, DisjointClasses and EquivalentClasses axioms are present in just under one third of the ontologies in the corpus. Moreover, SubClassOf axioms are present in large numbers when compared to EquivalentClasses axioms and DisjointClasses axioms—on average two orders of magnitude more. The picture for class constructors is similar: The dominant form of class constructor is Existential restriction (including ObjectSomeValuesFrom, DataSomeValuesFrom, ObjectHasValue, and DataHasValue). Nearly two thirds of ontologies contain axioms which use one or more type of Existential restriction. By contrast, Universal restrictions are used in 17% of ontologies and many of the other class constructors in fewer than 10% of ontologies. Furthermore, on average the occurrences of Existential restrictions are two orders of magnitude greater than the occurrences of Universal restrictions which are themselves on average two orders of magnitude greater than occurrences of all other types of class constructors. Finally, we observed that some ontologies contain MinCardinality 1 restrictions as a syntactic variant of Existential restrictions.

The stand-out axioms and class constructors from the BioPortal corpus are SubClassOf axioms and Existential restriction class expressions. We therefore decided to focus on these constructs in the simplified WebProtégé UI.

5 Evaluating Coverage

One of the features in the new WebProtégé is the ability of users to upload their ontologies to the WebProtégé server. Users created these ontologies with other tools or download them from the Web. Since we released WebProtégé, users have uploaded 230 ontologies to our server.[3] This corpus represents the naturally occurring ontologies that WebProtégé users want to work with. Therefore, this collection of ontologies offers a rich source of data that we can use to empirically drive forward the development of the tool. In this section, we analyse this corpus to assess how well the simple profile defined in Definition 2 covers the ontologies that people actually want to edit in WebProtégé. We then discuss how we can use this information to evolve WebProtégé in the future.

For the purposes of this evaluation we also examine two extensions of WPP. The first, WPP-Dis, extends WPP with DisjointClasses axioms and is defined in Definition 3, while the second, WPP-DisEq, extends WPP with DisjointClasses and EquivalentClasses axioms and is defined in Definition 4. The motivation for these extensions is to determine how many class frames are excluded from being represented in the simplified WebProtégé UI because of the fact that it does not display DisjointClasses axioms or EquivalentClasses axioms.

Definition 3 (WPP-Dis). *A WebProtégé Profile class-frame with DisjointClasses axioms (WPP-Dis) for a class A is defined as in Definition 2 but with the grammar augmented with the following production rule below, where the non-terminals* ***ClassFrameAxiom*** *and* ***ClassExpression*** *are specified in Definition 2.*

ClassFrameAxiom := *'DisjointClasses' '(' A* ***ClassExpression*** *')'*

[3] The WebProtégé privacy policy prevents us from making this corpus available in its raw form. Moreover, the analysis that we conducted looks at aggregated data and ontology constructs from a structural point. We do not critically examine any domain content of any projects.

Definition 4 (WPP-DisEq). *A WebProtégé Profile class-frame with* **DisjointClasses** *axioms and* **EquivalentClasses** *axioms (*WPP-DisEq*) for a class A is defined as in Definition 2 but with the grammar augmented with the following production rule, where the non-terminals* ***ClassFrameAxiom*** *and* ***ClassExpression*** *are specified in Definition 2.*

ClassFrameAxiom := *'DisjointClasses'* *'('* A ***ClassExpression*** *')'* |
'EquivalentClasses' *'('* A ***ClassExpression*** *')'*

Materials and Method On May 6th 2013 the version of WebProtégé hosted at Stanford contained 519[4] non-empty projects. Of these, 230 projects were created by users who uploaded their existing non-empty ontologies (the remaining projects were edited from scratch in WebProtégé). We parsed each non-empty ontology in the set of 230 using the OWL API version 3.4.3. We examined the classes in the signature of each ontology according to Definition 1 to determine which of them had WPP class frames satisfying Definition 2 (i.e. which of them can be represented by the simple UI). For each ontology, we measured the coverage in terms of the percentage of WPP, WPP-Dis and WPP-DisEq class frames.

Results Figure 3 shows a plot of class frames over the WebProtégé ontology corpus. Each bar represents one ontology (one project) with a non-empty class signature. The full length of a bar indicates the number of general class frames (Definition 1) in the ontology represented by that bar. Each bar is divided into four segments (note that zero size segments are not visible in the plot) representing the WPP class frames that satisfy Definition 2 (painted white); the WPP-Dis class frames that satisfy Definition 3 (painted grey); the WPP-DisEq class frames that satisfy Definition 4 (painted with a hatch effect); and the class frames that are neither WPP, WPP-Dis or WPP-DisEq frames (painted black). Figure 4 shows a plot representing the class frame coverage over the complete set of ontologies in the WebProtégé corpus. Each line represents class frames falling into the WPP (solid black), WPP-Dis (dashed black) and WPP-DisEq (dashed grey) profiles, with the plot showing the relationship between the number of ontologies and percentage of frames covered.

Analysis Broadly speaking, the simple UI in WebProtégé can represent the majority of class frames in the majority of ontologies in this corpus—there are 108 ontologies (or 47% of the corpus) for which it can present 100% of the class frames, and a further 12 ontologies (coming to just under 60% combined) for which it can present 90% of the class frames. Figure 4 plots the coverage of ontologies by the WPP as a black solid line. Each point on the line represents an ontology and the percentage of its terms covered by the profile. Combined,

[4] This number does not include a handful of projects created by the authors and colleagues at Stanford that we excluded from this analysis so as not to bias results. We also excluded several copies of the "pizza" ontology, which is a tutorial ontology containing most OWL 2 constructs.

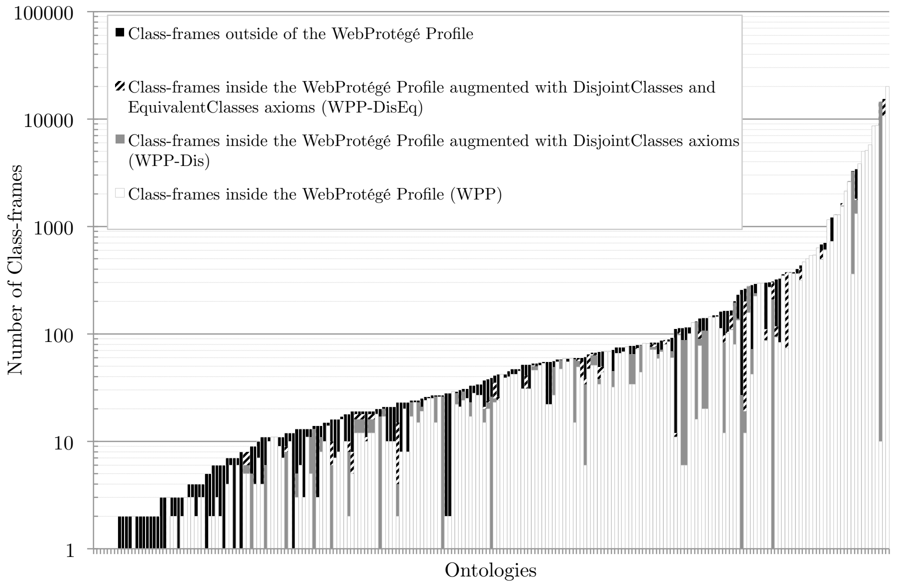


Fig. 3. WPP, WPP-Dis and WPP-DisEq class frames by ontology. Each bar represents one ontology with a signature size greater than zero. The total height of any given bar represents the total number of classes in the signature of the ontology.

there are 156 ontologies (just under 70% of the corpus) for which the WPP can capture 75% or more of the class frames in each ontology. These results are acceptable for a number of reasons: (1) the simple UI *completely* caters for a large fraction (roughly half) of the users that decided to edit their ontologies in WebProtégé—we expect a mix of novices and experts to use our tool and therefore the simple UI need not cater for everybody; (2) we do not expect the UI to cover the *whole* corpus—if it did, it would ultimately have to capture the full expressivity of OWL; and (3) it is conceivable that in collaborative settings there will be a cross-section of users, with less experienced users working with more experienced users. In these case, less experienced users may well prefer to edit the majority of the class frames in the ontology in the simple UI, while the more experienced users take care of class definitions that cannot be expressed in this UI.

At the lower end of the scale, there are three ontologies (1% of the corpus) for which the WPP *cannot represent any* class frames at all, and 38 ontologies (16% of the corpus) for which it can only represent 50% of class frames or less on average. A closure examination reveals that all of the ontologies that do not contain *any* class frames that are captured by the WPP, or ontologies that contain very low numbers of captured class frames, are like this because they contain

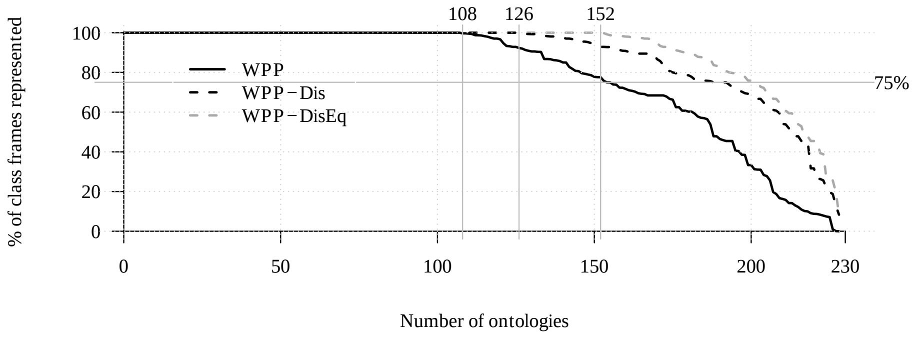


Fig. 4. Coverage of ontologies in the WebProtégé corpus. Each point on the X axis corresponds to an ontology; the Y axis plots the percentage of class frames that are covered in each profile. Each line represents a profile. The area under a line represents the number of class frames covered over the whole corpus by the profile represented by that line. As we extend the WPP profile to the WPP-Dis profile and then to the WPP-DisEq profile, the number of ontologies with 100% coverage increases.

DisjointClasses axioms. Looking at Figure 3, there are several long grey bands. These bands represent ontologies with large numbers class frames that are *not* captured by WPP (Definition 2) but *are* captured by WPP-Dis (Definition 3). In other words, they represent class frames which utilise DisjointClasses axioms. The effect of admitting DisjointClasses axioms to WPP is highlighted in the difference between plots in Figure 4. The plot shifts to the right, representing the ontologies for which there is 100% coverage, increasing by almost 20 ontologies with the addition of disjoint axioms. Put simply, admitting DisjointClasses axioms by supporting them in the simple UI would allow many more ontologies to be fully captured. In a similar vein, looking at the difference between the dashed black and dashed grey plots in Figure 4, it is clear that either including or excluding EquivalentClasses axioms has a noticeable effect on the number of ontologies all of whose class frames can be captured by a UI supporting these type of axioms—the number jumps from 126 (55%) to 152 (67%).

6 Evaluating Usability

In order to evaluate the usability of WebProtégé and to understand what the users like and dislike about the interface, we have conducted a survey among those users who had a chance to try the new WebProtégé design, either in its beta phase or after the official release.

Materials and Method We have designed the survey using SurveyMonkey®. The survey contained three types of questions: (1) qualification: the survey rules and the question asking respondents to confirm that they have had a chance to use the new version of WebProtégé. (2) usability: the questions about the user experience with the tool. (3) demographics: the questions about the user level of expertise and the type of projects that they were working on.

The survey included six usability questions [13] on a 5-point Likert scale (Figure 5) as well as free-text questions for feedback about the tool. Specifically, we asked what users liked about WebProtégé, what they felt needed to be improved, and what type of content they wanted to enter but could not. The last question in particular was designed to gain insight on what important OWL 2 constructs we failed to include in the WebProtégé Profile (WPP—Definition 2).

We emailed the survey link to all the users with the account on the WebProtégé server, to the Protégé support mailing lists, and posted the link on the Protégé social media channel. The survey was open for seven days. While the survey was completely anonymous, participants had the option of entering into a draw for a $25 gift card as a reward for their participation. Contact details for this were collected via a completely separate Web-form to preserve anonymity.

Results We received 55 responses from those who confirmed that they have used the new version of WebProtégé; 23 of the respondents chose to enter the draw for the gift card. Given that the WebProtégé change history lists contain changes or actions from distinct 288 users, our survey contains responses from 19% of the users who tried or used the system. The vast majority of respondents (90%) followed the link to the survey from the direct invitation email. Others followed the link in one of the Protégé mailing-list posts (6%), with the remainder using the links on Web sites. Among the respondents, 70% were from academia and 17% from industry, with the remainder from government, museums, and other organisations. We received responses from across the world, with the largest share of contributions from Europe (40%) and North America (40%).

As far as users' self-reported level of expertise with ontologies and OWL is concerned, on a 5-point Likert scale (1-Beginner and 5-Expert), the average expertise in ontologies was 2.96 (1:15%, 2:21%, 3:27%, 4:27% and 5:10%) and the average expertise in OWL was 2.7 (1:21%, 2:19%, 3:35%, 4:21% and 5:4%). All respondents have performed some content editing in WebProtégé, either editing an ontology (64%), uploading an ontology (57%), downloading an ontology (45%), defining sharing settings (38%), and other actions. 17% of the respondents participated in collaborative editing.

Figure 5 shows the distribution of answers to the usability questions on a 5-point Likert scale (1-Strongly disagree to 5-Strongly agree). Overall, 78% of the users agreed that they were satisfied with WebProtégé; 75% agreed that it was easy to use and 70% agreed that it was easy to learn. We have also looked separately at the results from the self-identified experts in ontology development and self-identified beginners. On all questions in Figure 5, experts were slightly more positive than the overall cohort. Beginners found collaborative features less useful than the overall group. In general, as can be seen from Figure 5 the responses for all questions are skewed towards "agree" and "strongly agree".

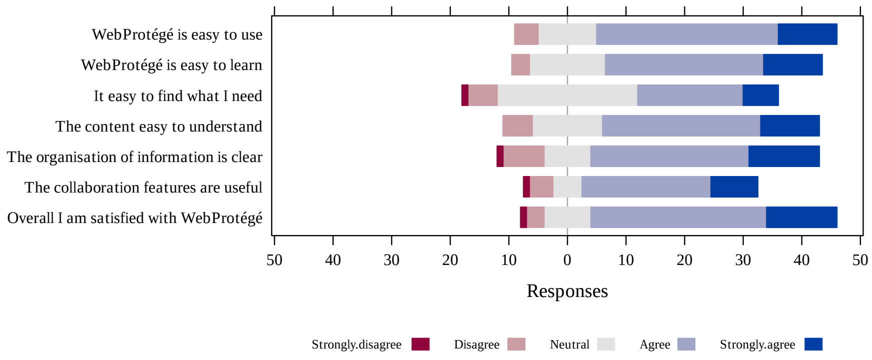


Fig. 5. A plot of responses to questions in the usability questionnaire. Blocks on the left of the centre-line (dark/light red) represent negative responses (strongly disagree/disagree). Blocks on the right of the centre-line (mid/dark blue) represent positive responses (agree/strongly agree). Blocks on the centre-line represent neutral responses. The size of each block is proportional to the number of responses.

We asked the survey respondents to identify specific content that they wanted to enter but were not able to enter in the simplified interface. Of the 55 respondents, one missed the ability to directly "create anonymous classes" and one wanted to "create logical expressions". The other 53 respondents did not indicate any specific constructs that they were not able to enter in the simplified UI.

Analysis While our survey results are limited to the early adopters of the tool the results are encouraging. The overall skew of the usability question responses towards "agree" and "strongly agree" indicates that users feel comfortable using the tool. The fact that only two users commented that they could not enter complex class expressions seems to indicate that users do not necessarily feel limited by the simple UI. Finally, given the mix of respondent expertise in ontologies and OWL, we believe that the simple UI might be capturing the best of both worlds: simple enough for novices to learn and to use, yet powerful enough for experts to do their job. Indeed, when using the interface ourselves to develop ontologies, we observed that we ourselves appreciated the many shortcuts that WebProtégé now provides as they made our work more efficient.

7 Discussion

In the past decade, researchers developed a number of Web-based ontology-development tools, such as OntoWiki [14], MoKi [15], Neologism [16], PoolParty [17], TopBraid EVN and others. Similarly, semantic wikis add semantic capabilities to traditional wikis. These semantic wikis [18] usually associate a Web page with a particular instance in the ontology, and the semantic Web annotations are converted into properties of that instance. Several works have proposed using controlled natural language to enter OWL constructs as a way of simplifying construction of OWL ontologies [19, 20]. These tools make a variety of trade-offs in terms of which constructs to present to the users. To the best of our knowledge, WebProtégé is the first web-based interface for ontology development designed empirically, based on a large ontology corpus.

We used one corpus—BioPortal—to design the interface. Our evaluation of this interface against a new corpus demonstrated the general validity of our approach. At the same time, it highlights two key types of axioms—disjoint classes axioms and equivalent classes axioms—that account for a notable fraction of this new corpus that cannot be represented in the WebProtégé profile. We are currently evaluating several approaches to extend the expressive power of the user interface. First, we can expand the default UI to account for these types of axioms. We will evaluate how much it affects the simplicity and usability of the interface: there is a danger that adding more expressive power will clutter the interface and take away what the users currently like about it. Second, we can design a second preconfigured interface, which will be geared towards the users who are more experienced with OWL and will provide greater expressive power. We plan to investigate whether or not we can limit the default interface to a single interface that satisfies all our users (something that our survey indicates might be possible) or whether we need multiple configurations. Finally, we can leave it up to the users to configure the WebProtégé UI to satisfy their needs. One of the key features of WebProtégé is that it allows users to custom-tailor their interface, choosing which components they see in the class definitions, and which widgets they use from each component. For instance, a user that needs to write complex OWL class expressions that are not supported by the simple UI can enable a UI component that looks similar to the class description editor in the desktop version of Protégé.

8 Conclusions

In this paper, we presented the new version of WebProtégé, a web-based OWL ontology editor with an empirically designed simple user interface. This user interface accounts for a large fraction of ontologies and class frames in two large ontology corpora. Yet, a mix of beginner and expert users perceive it as being both easy to use and easy to learn and they are satisfied with the interface. Our data shows a significant community uptake. These results point to a novel way to address the complexity of ontology development through an iterative process that relies on empirical data and feedback from the user community.

Acknowledgements

This work was supported by grants GM086587 and GM103316 from the National Institute of General Medical Sciences at the United States National Institute of Health. We are indebted to all the Protégé users for their continuous feedback and support.

References

1. Gibson, A., Wolstencroft, K., Stevens, R.: Promotion of ontological comprehension: Exposing terms and metadata with Web 2.0. In: Workshop on Social and Collaborative Construction of Structured Knowledge at WWW 2007. (2007)
2. Nixon, L., García-Castro, R., Wrigley, S., Yatskevich, M., Santos, C.T.D., Cabral, L.: The state of semantic technology today - overview of the first SEALS evaluation campaigns. In: 7th Int. Conf. on Semantic Systems. (2011)
3. Horridge, M., Bechhofer, S.: The OWL API: A Java API for OWL ontologies. Semantic Web Journal **2**(1) (2011) 11–21
4. Musen, M.A., Noy, N.F., Shah, N.H., Whetzel, P.L., Chute, C.G., Storey, M.A., Smith, B., the NCBO team: The National Center for Biomedical Ontology. Journal of American Medical Informatics Association **19** (2012) 190–195
5. Whetzel, P.L., Noy, N.F., Shah, N.H., Alexander, P.R., Nyulas, C.I., Tudorache, T., Musen, M.A.: BioPortal: Enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. Nucleic Acids Research (NAR) **39**(Web Server issue) (2011) W541–5
6. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web Ontology Language structural specification and functional style syntax. W3C Recommendation, W3C – World Wide Web Consortium (2009)
7. Mungall, C.: OBO Flat File Format 1.4 syntax and semantics (2011)
8. Golbreich, C., Horridge, M., Horrocks, I., Motik, B., Shearer, R.: OBO and OWL: Leveraging semantic web technologies for the life sciences. In: The International Semantic Web Conference. (2007)
9. Tudorache, T., Nyulas, C., Noy, N.F., Musen, M.A.: WebProtégé: A collaborative ontology editor and knowledge acquisition tool for the web. Semantic Web Journal **4**(1) (2013)
10. Corcho, Ó., Roussey, C.: OnlynessIsLoneliness (oil). In: Proceedings of the Workshop on Ontology Patterns (WOP 2009). (2009)
11. Rector, A.L., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., Wroe, C.: OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns. In: EKAW 2004. (2004) 63–81
12. Horridge, M., Parsia, B., Sattler, U.: The state of biomedical ontologies. In: BioOntologies 2011 Co-Located with ISMB 2011. (2011)
13. Nielsen, J.: Usability Engineering. Academic Press/Morgan Kaufmann (1994)
14. Auer, S., Dietzold, S., Riechert, T.: OntoWiki–a tool for social, semantic collaboration. In: ISWC 2006. (2006)
15. Ghidini, C., Kump, B., Lindstaedt, S., Mahbub, N., Pammer, V., Rospocher, M., Serafini, L.: MoKi: The enterprise modelling wiki. In: ESWC-2009. (2009)
16. Basca, C., Corlosquet, S., Cyganiak, R., Fernández, S., Schandl, T.: Neologism: Easy vocabulary publishing. In: Workshop on Scripting for the Semantic Web (ESWC-2008). (2008)
17. Schandl, T., Blumauer, A.: Poolparty: Skos thesaurus management utilizing linked data. In: ESWC 2010. (2010)
18. Krotzsch, M., Vrandecic, D., Volkel, M.: Semantic MediaWiki. In: ISWC 06. (2006)
19. Hart, G., Johnson, M., Dolbear, C.: Rabbit: Developing a control natural language for authoring ontologies. In: ESWC 2008. (2008)
20. Power, R.: OWL simplified english: A finite-state language for ontology editing. In Kuhn, T., Fuchs, N., eds.: Controlled Natural Language. Volume 7427. Springer (2012)