Observations in collaborative ontology editing using Collaborative Protégé

Daniel Schober[1,2], James Malone[2], Robert Stevens[3]

[1]Institute for Medical Biometry and Medical Informatics, University Clinic, Freiburg, Germany, [2]European Bioinformatics Institute, Cambridge, UK, [3]Manchester School of Computer Sciences, Manchester, UK

Abstract. Creation of an ontology according to some common plan is best accomplished collaboratively. This is sometimes contradicted by the distribution of the ontology's developers. An obvious solution therefore is to build collaboration into ontology development tools. Such support necessarily includes both the technical means to perform editing operations upon an ontology, but also support for the communication that makes collaboration such a vital part of much ontology development. To investigate the distributed, collaborative ontology engineering process and the corresponding capabilities of the Collaborative Protege 3 (CP) tool, members of the OntoGenesis network came together and enriched the Ontology of Biomedical Investigations (OBI) with new content. The communications and interactions of the participants with each other, directly or through the tool, were tracked and analyzed. Our initial analysis of the degree to which this new tool fulfills the practical requirements of collaborative ontology engineering suggests the approach is promising. We present some observations and recommendations for CP based upon this experience.

Keywords: Ontology, Collaborative Protégé, concurrent editing, requirement analysis

1 Introduction

Engineering ontologies that are representative of a community consensus is of great interest to those working in bioinformatics. Yet, the process of developing such ontologies often requires collaboration by many people in distributed geographical regions. There are a number of important requirements for ontology development tools that cope with the contradiction of the need for close collaboration and the distribution of developers [1]. The following ones were used for a large survey on collaborative ontology engineering tools [2]:

- Concurrent ontology editing, the ability for multiple edits to be made to the ontology at a single time and from different computers.
- Tracking annotations associated with corresponding representational units (RUs).

- Tracking annotations associated with actions on ontology change, such as deletions, axiom and annotation edits.
- A manageable mechanism for discussion threads and instant messaging for online editors that satisfy the need for communication between ontology developers so necessary for collaboration.

The new Collaborative Protégé (CP) plugin [3] for the widely used open-source ontology editing tool Protégé 3, aims to support the above features. It enables concurrent editing of a single OWL file and also features notes on RUs, a change tracking log for RUs (such as class edits), a discussion thread and an instant messaging client for real time chat. The tool captures changes, annotations and discussions as instances of an integrated *Change and Annotation Ontology (ChAO)*, thereby providing a useful audit trail of edits and decision making [4]. Although user surveys indicate that ontologists with a software engineering background prefer asynchronous ontology editing [5], and although methodologies emerge that handle ontology editing via concurrent versioning systems (CVS) [1, 6], their setup and usage is generally more complex than concurrent real-time editing and ontologists with a life science background often prefer synchronous concurrent editing [5].
CP appears an appropriate choice for an evaluation of collaborative ontology engineering, because it is based on a widely used platform and an evaluation was explicitly requested by several groups [2, 7]. In this paper we present an initial investigation into CP's practical use.

2 Materials and Methods

Thirteen members of the OntoGenesis Network came together at the European Bioinformatics Institute (EBI) for the 7th OntoGenesis Meeting (Dec. 15th and 16th 2008, website: http://ontogenesis.ontonet.org/moin/NetworkMeeting7). The instrument branch of the ontology of biomedical investigation (OBI, http://obi.sourceforge.net), an OWL-DL ontology for the annotation of the biomedical laboratory workflow, was chosen as the topic for this study. The aim was to enrich the ontology with new classes and relations needed to describe instruments. The instrument branch was chosen because it represents an area of daily experience upon which a broad range of biologists, such as is present in the OntoGenesis Network, have something valid to contribute; all biologists use devices of some kind. The OBI.owl file was populated with new device classes and functions, a) coming from the domains of the OntoGenesis members, and, b) as taken from a list provided by the Metabolomics Standard Initiative (http://msi-ontology.sourceforge.net/). The development followed the methodology adopted by the OBI developers (http://obi.sourceforge.net/ontologyInformation/index.php#designPrinciples).
Numeric data were gathered from the resulting knowledge bases and put into spreadsheets, which can be found at http://www.imbi.uni-freiburg.de/~schober/CPEvalStats_v8.xls and which were used to generate the diagrams in this paper. Since this was a more informal, observational study and a full statistical analysis and evaluation is beyond its scope, some of the numbers and metrics generated have to be interpreted in that context. Our study was designed to

examine how well the CP tool (v. 3.4 beta) supported the basic requirements outlined above. In general, we wanted to explore the technical aspects of concurrent editing of one ontology and the support for communication between a number of ontologists working concurrently. Thus our study was based upon the following set of tasks:

Familiarization*:* Users had an initial familiarization with Collaborative Protégé 3.4, its GUI and collaborative features.
***Ad hoc* additions**: Development of attendee's own lists of devices and concomitant functions. This essentially required the addition of new classes as children of the OBI device class and the OBI function class. This also meant that there was a possibility of duplication, i.e. addition of the same device by 2 different editors, as the edits were made concurrently. In this session we wanted to know how such conflicts were resolved.
Controlled additions*:* Placement of selected device classes from the MSI term list into OBI. The appropriate metadata required by OBI were also added. In this session we wanted to know how agreement (on subsets) is coordinated.
'Agent Provocateur'*:* During a specified time period known only to organizer, an *Agent Provocateur* added conflicting and deliberately incorrect content to the ontology. This was used to assess the transparency of the changes to the other online editors. In this session we wanted to know whether faults and nonsense edits were detected by other editors.
Controlled Communication*:* Communication was restricted to specified channels during each editing session in order to evaluate CP’s ability to foster communication, i.e. via notes, discussion threads and chat. In this session we wanted to know how CP fosters problem solving in communication.

To simulate a distributed collaborative setting, the group was divided into subgroups:
Single group (n=13): Familiarization with the tool.
Two groups (n=6): *Ad hoc* additions were made, where editors were able to add and edit classes as they saw appropriate.
Four groups (n=2): The pairs tackled different subsets of the MSI device term list. Each pair picked new terms from the list and added them to OBI with discussion. Then the results of the pairs were reviewed and commented upon by other pairs adding annotations.
Single group (n=13): More MSI terms were added by the whole group and the *Agent Provocateur* user was deployed, provoking discussions, which was in turn used to test the different communication channels provided by Protégé:

- First, the chat was used to comment, annotate and discuss additions.
- Then they were discussed by voice only.
- Then by chat and voice together.

4 Results

CP installation and setup
From a technical perspective, the setup was quick (30 minutes). The installation guide was clear and easy to follow. Only minor updating of the installation guideline was required. CP can be configured with user name and password. We provided the same passwords for all clients, which led to some abuse of the ability to log on as other users (three cases of 'identity fraud'). The normal Protege plugins appear and work in CP, some however need to be installed on the server as well as on the clients in order to correctly work. Since most are part of the standard distribution, these are all

directly usable. The CP tabs can be configured according to a project's needs and the default set of annotation properties created on class creation could conveniently be configured according to OBI policies. CP copes with more complicated ontology setups. When configured correctly, it displays the rdfs:labels instead of the numeric IDs in the class hierarchy, as specified in the pprj file.

Ontology Editing Statistics

The annotation_OBI.rdf file grew by 46,1 % per day, so nearly doubled in size during the 2 day meeting, indicating linear growth and no performance problems when projected into the future (see Fig. 1). The OBI file grew 4.3% over the meeting course, whereby the increase in added defined classes (5 abs, 10.2%) was nearly double that of primitive classes (4.8%), but this relatively high number is also due to the fact that the ontogenesis participants were all experienced when it comes to the creation of logical class definitions, e.g. compared with the whole OBI dev group, where the number of primitive classes grows more rapidly than that of defined classes. The large 'siblings (max)' value is due to the OBI class obsolescence policy, that collects all deleted classes as siblings in an '_obsoleted' helper class.

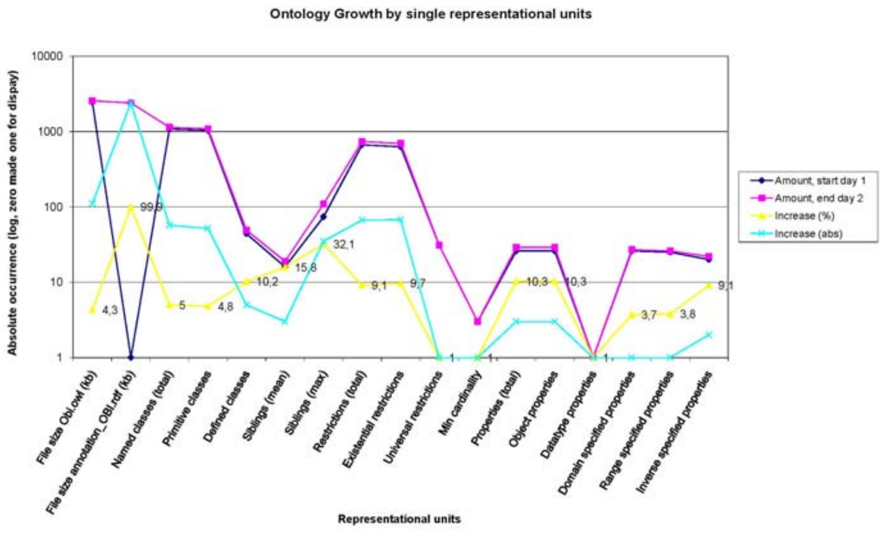


Fig. 1: This diagram shows the development of the project metrics over time (RU and RA). The absolute values, as computed from the OBI.owl file at the start and at the end of the meeting are given, as well as the increase in the number of individual entities (abs and %).

Three new object properties were created during the meeting. These were used in a total of 68 new existential restrictions (9.7% increase).

In general, we discovered that the large differences in overall activity appear to be the result of experience, personality-structure, and confidence level of the users. The apparent general trend was that people that chat a lot also made more changes on the ontology. The data show different users working on different parts of the ontology and on different RU types (E.g. user 7 worked on relations, user 5 on annotations, see Fig. 2).

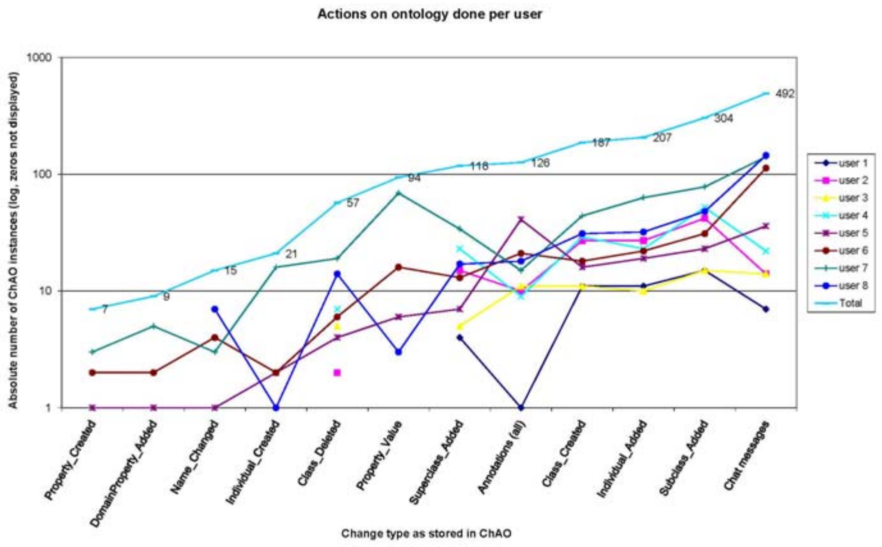


Fig. 2: Changes done on the ontology. This diagram illustrates the distribution of changes/actions done on the ontology by individual users (stored as ChAO instances).

The quality of the changes has not been evaluated at this time. Many classes were edited by several editors, with an average of two editors per class. Changed classes: 13, (removed and added restrictions, changed superclasses, changed from primitive to defined, added annotations). The ratio of created to deleted classes was 2,2 for user 8 and 3, 2,3 for user 7, 3 for user 6, 4 for user 5, 4 for user 4 and 13,5 for user 2. We hypothesize that the ratio is smaller in users that generally made more changes than in more 'careful' users.

Features requested

The lack of a RU and module locking mechanism meant there was no way to temporarily prevent others from altering classes that have a logical impact on the class under current definition [5]. If a highly nested class description is created, it is difficult to get it right, unless others are prevented from changing something higher up in the hierarchy that will contradict the definition currently worked upon. Another method would be to simply highlight areas that are currently worked on according to a colour scheme identifying the users, who then could resolve this by chat.

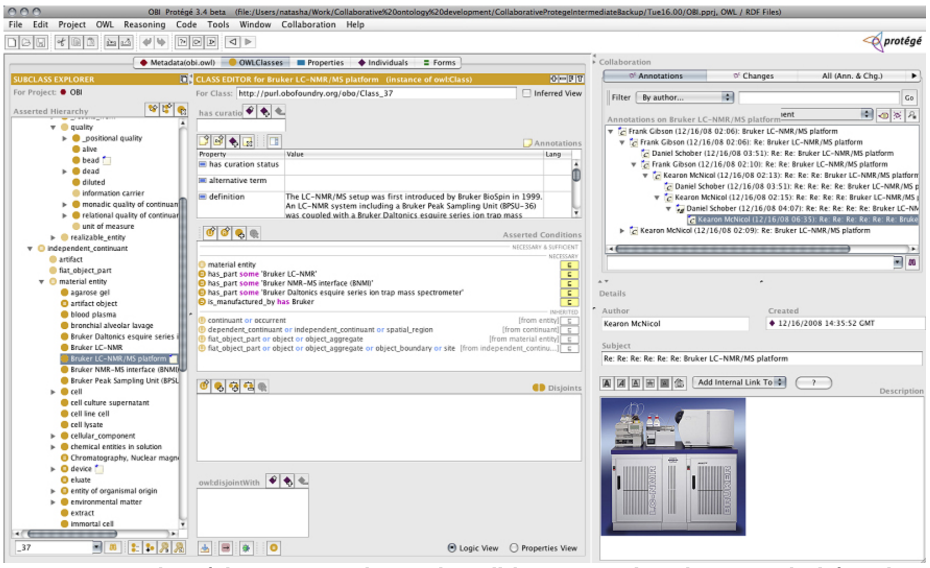
Checking for duplicate class and property labels and notification of the users would be useful. If two users added the same class concurrently, there was no notification after the duplication had occurred.

A priority has to be the undoing of class deletions, because this can occur accidentally very easily in Protégé, e.g. by a single wrong click on the delete button or by accidentally moving classes. The deletion of a class removes all its subclasses and this cannot be undone. A roll back function would aid in conflict resolution and would lead to safer editing. Undo/redo functionalities would be another feature to help users to prevent conflicts. Some non-deprecated properties were found to be sub-properties

of deprecated properties, which seemed odd. Since currently there is also no global change list, it is impossible to see changes and annotations on deleted entities. If a parent class is deleted, all its annotations disappear, including all children. The annotations will still be there, but since the association to the annotated RU is done via the ID only, without the label it is difficult to know what was annotated.
Subscription and notification of changes was requested, where users subscribe to certain areas of interest within the ontology and are then notified of any changes that occur to those areas. Being notified of changes chosen by a user, such as discussion threads or certain RUs, would help to stay up to date and proceed faster in conflict resolution. For example warnings and alerts could be passed to subsets of users to prevent duplicate or contradicting editings. A 'change view' on selected items that are on a watch list would help users to keep track of recent developments in their areas of interest or responsibility. A feed of all classes could be used to notify developers to subscribed classes. For the annotation flag in the class hierarchy it would be practical to see when someone added some new annotation, e.g. the annotation flag should then get an exclamation mark, or blink, or should display an analog bar that indicates the number of attached annotations (a measure of *topic-hotness*).

Versioning
A side benefit of using a real time collaborative approach is that complicated versioning strategies are not needed: SVN change track and diff functions are not feasible for OWL files and practical solutions are only beginning to emerge [6]. Using SVN the threshold to do minor changes can be increased on the user side, because a complicated merge back and conflict resolution needs to be carried out on the whole artifact level, even when logically non-conflicting changes were made. However, even when SVN is used, the change track captured in the ChAO knowledge base (KB), can be copied and distributed in some SVN log after updating OWL files. One drawback here is that small changes result in a textual information overkill: for a human readable change history, the tool should just state ‘class x was moved from A to B’, instead of listing all involved quantum changes, e.g. 'class x was deleted from A', 'class x was created under B', …. Users would like the changes to be described at a higher level abstraction, rather than being overly granular.

Annotations on RUs with entity notes
Due to its abundant connotation with OWL annotation properties, the term "annotations" as used in the CP GUI caused some initial confusion. Consequently, the "Annotations Tab" has now been re-labeled to "Entity Notes" which is clearer and more specific. "Discussion Threats" has been renamed to "Ontology notes" correspondingly. Unfortunately these name changes are now out of sync with the nomenclature used in the ChAO ontology.
Each annotation has a freetext subject field to fill in as well as its freetext value (Fig.3). For the majority of small annotations, it turned out that people did not use the subject heading, potentially because they felt to provide an annotation type, subject heading and value for small annotations was too great an effort. Seeing the annotations in a table view, e.g. sorted according to type, subject and value would make viewing easier. Axiom annotations, as being currently investigated for OWL2, were also requested by some users.



OBI Protégé 3.4 beta (file:/Users/natasha/Work/Collaborative%20ontology%20development/CollaborativeProtegeIntermediateBackup/Tue16.00/OBI.pprj, OWL / RDF Files)

SUBCLASS EXPLORER

For Project: OBI

Asserted Hierarchy

```
    quality
      _positional quality
      alive
      bead
      dead
      diluted
      information carrier
      monadic quality of continuan
      relational quality of continuar
      unit of measure
    realizable_entity
  independent_continuant
    artifact
    fiat_object_part
    material entity
      agarose gel
      artifact object
      blood plasma
      bronchial alveolar lavage
      Bruker Daltonics esquire series i
      Bruker LC-NMR
      Bruker LC-NMR/MS platform
      Bruker NMR-MS interface (BNMI
      Bruker Peak Sampling Unit (BPSU
      cell
      cell culture supernatant
      cell line cell
      cell lysate
      cellular_component
      chemical entities in solution
      Chromatography, Nuclear magn
      device
      eluate
      entity of organismal origin
      environmental matter
      extract
      immortal cell
```

_37

CLASS EDITOR for Bruker LC-NMR/MS platform (instance of owl:Class)

For Class: http://purl.obofoundry.org/obo/Class_37 Inferred View

has curatio

Annotations

Property	Value	Lang
has curation status		
alternative term		
definition	The LC-NMR/MS setup was first introduced by Bruker BioSpin in 1999. An LC-NMR system including a Bruker Peak Sampling Unit (BPSU-36) was coupled with a Bruker Daltonics esquire series ion trap mass	

Asserted Conditions

NECESSARY & SUFFICIENT

NECESSARY

material entity
has_part some 'Bruker LC-NMR'
has_part some 'Bruker NMR-MS interface (BNMI)'
has_part some 'Bruker Daltonics esquire series ion trap mass spectrometer'
is_manufactured_by has Bruker

INHERITED

continuant or occurrent [from entity]
dependent_continuant or independent_continuant or spatial_region [from continuant]
fiat_object_part or object or object_aggregate [from material entity]
fiat_object_part or object or object_aggregate or object_boundary or site [from independent_continu...]

Disjoints

owl:disjointWith

Collaboration

Filter By author...

Annotations on Bruker LC-NMR/MS platform

```
Frank Gibson (12/16/08 02:06): Bruker LC-NMR/MS platform
  Frank Gibson (12/16/08 02:06): Re: Bruker LC-NMR/MS platform
    Daniel Schober (12/16/08 03:51): Re: Re: Bruker LC-NMR/MS platform
    Frank Gibson (12/16/08 02:10): Re: Re: Bruker LC-NMR/MS platform
      Kearon McNicol (12/16/08 02:13): Re: Re: Re: Bruker LC-NMR/MS platform
        Daniel Schober (12/16/08 03:51): Re: Re: Re: Re: Bruker LC-NMR/MS p
        Kearon McNicol (12/16/08 02:15): Re: Re: Re: Re: Bruker LC-NMR/MS p
          Daniel Schober (12/16/08 04:07): Re: Re: Re: Re: Re: Bruker LC-NM
            Kearon McNicol (12/16/08 06:35): Re: Re: Re: Re: Re: Re: Bruke
  Kearon McNicol (12/16/08 02:09): Re: Bruker LC-NMR/MS platform
```

Details

Author: Kearon McNicol

Created: 12/16/2008 14:35:52 GMT

Subject: Re: Re: Re: Re: Re: Re: Bruker LC-NMR/MS platform

Description

Fig. 3: A screenshot of the CP GUI is shown. The Collaboration Panel can be seen at the left, with an annotated class highlighted and marked in the hierarchy. The middle section contains the usual Protégé Class and Restrictions Editor. A (deep) discussion thread on the Chromatography Platform is shown in the CP panel to the right, including a picture in an annotation.

The group observed that, to avoid information overload and to keep quality up, users should be allowed to remove their unintended annotations e.g. for the first 5 min of their creation.

Numbers and kinds of annotations made

By far the most abundant annotation/note was the comment (89 abs, assumingly due to its 'default' setting). The distributions of comments over classes followed a power law distribution: A few classes had a large number of annotations (> 15 each) and a large number of classes had < 4 annotations/comments. For two users (1 and 4) the comment was the only way they annotated entities, ignoring all other annotation types. Then followed Advices (14) and AgreeDisagreeVotes (12). There were a few AgreeDisagreeVoteProposals and Questions (6 each). Example and Explanation were seldom used and by more experienced 'senior' users. In general the overall distribution of annotations over the annotation types was highest among the latter. Unfortunately, no annotations were made on changes of the ontology, all annotations referred to RUs in the ontology. This hints at low perception of this possibility in the user group, which appears to be mainly due to the hidden Change ontology. No SimpleProposal, FiveStarProposal, FiveStarVote and seeAlso had been carried out.

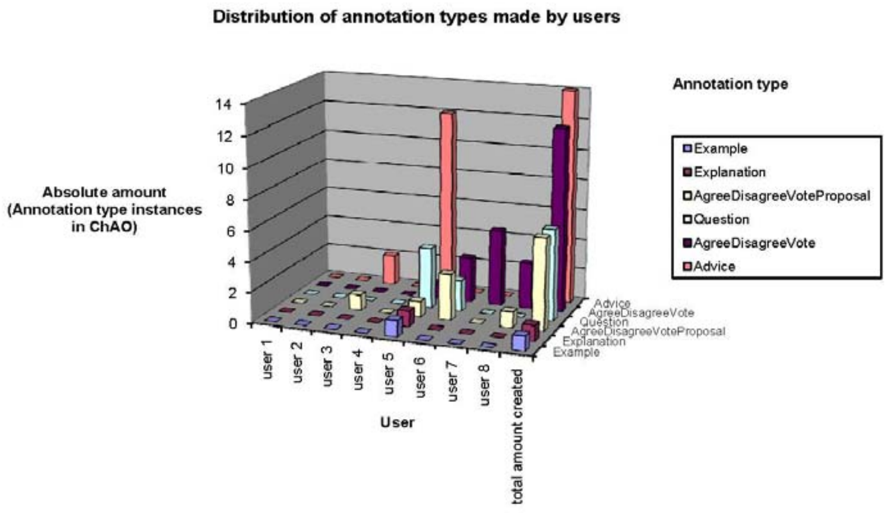
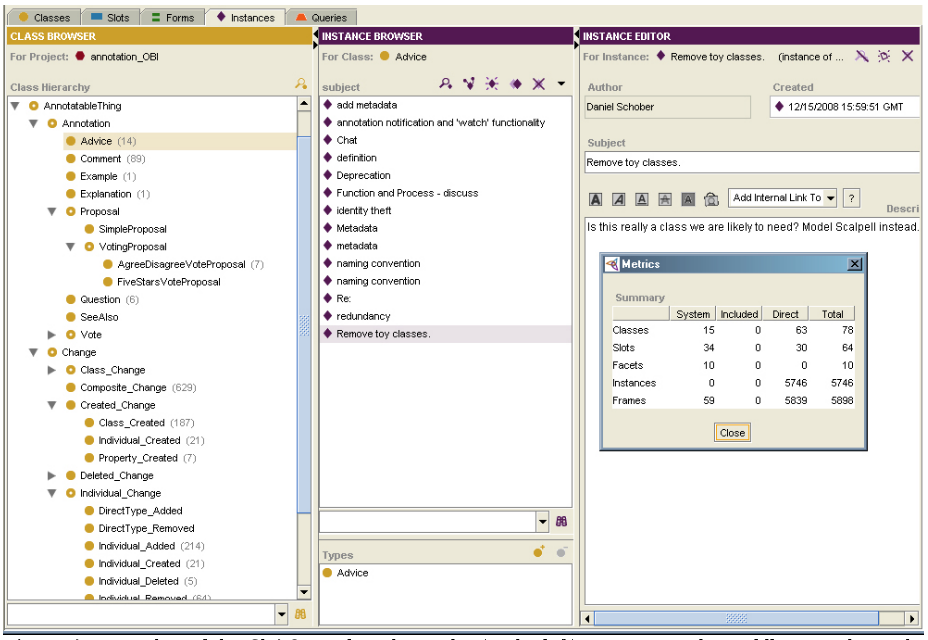


Fig. 4: This diagram shows the distribution of annotation types used by individual users, sorted according frequency (total, from last annotation_obi InstanceTab). The highly abundant default ‘comment’ had been omitted here to keep the axes in a reasonable scale and keep the diagram readable.

There was no power law distribution for the number of comments done per person: most users made around 10 comments. Patterns in annotation behavior could be used to infer roles of users like 'commenter', 'chatter' or 'editor'. User 5, made twice as many comments as others, often creating tasks for others like 'add metadata', 'remove redundancy', hinting at a 'moderator role'. This role is further indicated by the use of more granular annotation types, e.g. ‘advice’, ‘explanation’ as indicated in Fig. 4. Regarding participants roles, it would be interesting to interview the participants on their self-assessment or try to find motivations behind certain actions, e.g. competition, altruism, narcissism, self-interest.
The mean depth of discussion threads created was 2-3, the max thread depth was 5 (responses). Regarding the chats, only 12 messages used internal hyperlinks (4 on day 1, 8 on day 2). In harmony with findings of [8], issues discussed via chat were mainly about what to work upon next, modeling issues and new feature requests or their implementation. Experimental helperclasses were created: '_Kearon's collect devices by function classes', 'Frank's new meaning of function' and 'asserted_gibbon_disco', but only one user adhered to the OBI policy to indicate such play-classes with the underscore prefix (see first example provided).
On the positive side, we note that in cases where the present (meta-) annotations are not sufficiently granular, the annotation types in CP’s underlying ChAO (Fig.5) can be expanded with new annotation types that suit special projects needs and evaluation approaches.



CLASS BROWSER

For Project: annotation_OBI

Class Hierarchy

- AnnotatableThing
  - Annotation
    - Advice (14)
    - Comment (89)
    - Example (1)
    - Explanation (1)
    - Proposal
      - SimpleProposal
      - VotingProposal
        - AgreeDisagreeVoteProposal (7)
        - FiveStarsVoteProposal
    - Question (6)
    - SeeAlso
    - Vote
  - Change
    - Class_Change
    - Composite_Change (629)
    - Created_Change
      - Class_Created (187)
      - Individual_Created (21)
      - Property_Created (7)
    - Deleted_Change
    - Individual_Change
      - DirectType_Added
      - DirectType_Removed
      - Individual_Added (214)
      - Individual_Created (21)
      - Individual_Deleted (5)
      - Individual_Removed (64)

INSTANCE BROWSER

For Class: Advice

subject

- add metadata
- annotation notification and 'watch' functionality
- Chat
- definition
- Deprecation
- Function and Process - discuss
- identity theft
- Metadata
- metadata
- naming convention
- naming convention
- Re:
- redundancy
- Remove toy classes.

Types

- Advice

INSTANCE EDITOR

For Instance: Remove toy classes. (instance of ...

Author: Daniel Schober

Created: 12/15/2008 15:59:51 GMT

Subject: Remove toy classes.

Descri

Is this really a class we are likely to need? Model Scalpell instead.

Metrics

Summary

	System	Included	Direct	Total
Classes	15	0	63	78
Slots	34	0	30	64
Facets	10	0	0	10
Instances	0	0	5746	5746
Frames	59	0	5839	5898

Fig. 5: A screenshot of the ChAO ontology hierarchy (to the left) in Protege. The middle pane shows the list of "Advice" instances, with the last one (an advice to remove toy classes) marked and shown in detail to the right. All changes made to the ontology, get a user/date stamp and are stored as ChAO instances. The inset box displays the overall metrix of the ChAO KB associated with the OBI project.

It is possible to filter annotations by author, annotation text, annotation type or by creation date, alone or in logical combinations. Local metadata schemes, e.g. certain OBI annotation properties like has_curation_status or definition_source, can be queried for by the queries tab.

Communication

During the GUI familiarization period, threads and notes were mis-used for chats and *vice versa*, the latter being due to the chats' instant visibility and notification. Once a topic had started, it seemed to be difficult to find a cut off, when to move from a chat into an RU note or thread and *vice versa*. A good example of the consequence of not using the right modality for annotations was, when a participant warned the group about an obsolete property (is_device_for) in the threads and not in the more appropriate entity note for the object property RU. As a consequence it was found that despite a warning being issued, people used this obsolete property.

All messages are stored with message type, username and date stamp, and hence can be evaluated in detail, e.g. via SPARQL or the Protégé Queries Tab. Chats are stored in a separate chat project KB, but get overridden with each server re-start. Chats were used for general acute issues and planning, e.g. "vote being held on @'http://purl.obofoundry.org/obo/Class_44'. Chats were requested to be linked with specific RUs and axioms to aid a more immediate and direct conflict resolution and not overload the (persistent) entity notes. A closed 'retreat room' was desired as well

as a filter function on user names to enable to see only the chats of certain people or on particular ontology fragments.
The integration of emoticons in text fields could increase transmittance of pragmatic aspects of communications and would aid in the prevention of tensions on a sociological level, i.e. allowing irony to be expressed.
Integrated voting on change issues, proved to be not fully implemented, but was needed by users. A mechanism that changes the ontology automatically could increase KB development time and could be implemented using ChAO information and formalized voting outcomes.
Issue tracker functions were requested, i.e. a scratch pad or todo list that can be worked through and 'checked', e.g. indicating a proposed plan and what has already been resolved at a certain time point. For example, when people add new classes from a spreadsheet or source forge term tracker, they should have a checklist that indicates those classes with resolved issues.

Performance
Since Collaborative Protégé (CP) is an extension of the standard Protégé 3 Ontology editor, most users were quickly familiar with the CP GUI. Initial loading of the whole OBI file took 2 minutes on a Pentium 4, 512MB machine. General performance could be increased by enlarging the Heap Size and by removing concurrent projects from the metaproject. The performance can be increased and the risk of data loss is further minimized, when the owl file is converted into a relational database using the provided database backend. RAM requirements are less and dynamic loading will increase overall performance (also allows rollback).
Overall, the performance of CP was very usable and much can be done with configuration for further optimization. In large artifacts, expanding the full class hierarchy at once for the first time in one client can take a long time (ca. 20 sec in our setup). Also opening a class with many direct subclasses for the first time will slow down and initially impair performance.
Discussion and annotation update throughout the clients was so slow, that it led people to use the chat functionality, which was updated and immediately visible. To see an Annotation update, people needed to change a frame and only then was the GUI updated. This bug has since been rectified by the protégé team.

Conclusions

In this investigation, a realistic collaborative ontology building session was created using CP and its features were thoroughly tested in a series of informal experiments. Of the four typical phases of collaborative approaches to ontology design [9], CP mainly supports the *iterative improvement phase* that populates and revises the ontology until consensus is reached. In this respect, the setup of working with, and expanding an already existent ontology was sensible.
Although some caveats persist and some requirements could not be fulfilled at this time, it became clear that the CP tool is now in an advanced stage and can be used in practice with sufficient stability. It copes with complicated setups and is flexible

enough to allow for corresponding adjustments. The integrated ChAO metadata set provides an easy to use but flexible and expandable way of adding multimodal metadata to a knowledgebase in construction, enabling granular annotation types customized according to ones project's special needs and evaluation approaches. Although there are still some collaborative features missing [7] the way the tool leverages on ChAO makes integration of advanced communication techniques e.g. argumentation based agreement finding [10] a feasible task. The binding of discussions directly to the RUs they refer to saves users from 'issue archaeology', the futile searching in email-lists, teleconference notes, wikis, and term trackers [11].
From an overall technical point of view, collaborative ontology building was relatively trouble free. The main area for improvement comes from the need for more sophisticated communication mechanisms. In editing, a mechanism for conflict resolution, e.g. 'undo/redo' is needed, as well as some transaction management. Although crucial to editing in a collaborative, concurrent, real-time fashion, this is not presently available in CP. Some enhancement of editing functionality and the addition of notifications on changes to notes and threads was deemed necessary. The addition of chats to specific RUs and for specific groups would further enhance annotation traceability. In all, CP as it stands now is already usable as a collaborative tool that we can recommend, as it fulfils the three most requested characteristics of multi-user OE, as surveyed in [5], namely simplicity, provenance capture and flexibility. While for community driven evolution of artifacts of low semantic complexity editors of the Web 2.0 realm might be better suited [12], CP supports the creation of description logics-based formal ontologies more completely.
Our analysis provoked much feedback to the CP developers, and will be valuable for the CP version of Protégé 4 that is currently in preparation. Further use of CP in more controlled settings will enable us to acquire further insights into the process of tool-based collaborative ontology building, in collaboration modes other than concurrent, and such findings will be fed back to tool developers.

Acknowledgments

Thanks go to Tania Tudorache, Timothy Redmond and Natasha Noy from the Stanford Protégé team and James Watson from the EBI teaching facilities. Further thanks go to all the ontogenesis network participants, and the OBI Consortium for providing the merged OBI.owl file. DS is funded by the DebugIT project of the EU 7th FP (ICT-2007.5.2-217139). DS likes to thank his former and current employers for making this work possible in-between affiliation change. JM is funded by the EMERALD EU project number LSHG-CT-2006-037686. The OntoGenesis Network is funded by EPSRC grant EP/E021352/1.

References

1. Bao J, Caragea D, Honavar V: **Towards collaborative environments for ontology construction and sharing.** In *International Symposium on Collaborative Technologies and Systems (CTS 2006).* IEEE Press; 2006: 99-108.
2. Noy N, Chugh A, Alani H: **The CKC Challenge: Exploring Tools for Collaborative Knowledge Construction.** In *BMIR-2007*. 2007: 1260.

3. Tudorache T, Noy N, Tu S, Musen M: **Supporting collaborative ontology development in Protege.** In *Seventh International Semantic Web Conference; Karlsruhe, Germany*. Springer; 2008
4. Hartel FW, Fragoso G, Ong Kl, Dionne R: **Enhancing quality of retrieval through concept edit history.** *AMIA Annu Symp Proc* 2003**:**279-283.
5. Seidenberg J, Rector AL: **The state of multi-user ontology engineering.** In *Proceedings of the 2nd International Workshop on Modular Ontologies*. 2007
6. Jimenez-Ruiz E, Cuenca Grau B, Horrocks I, Berlang R: **Building Ontologies Collaboratively Using ContentCVS. Proceedings of the 22nd International Workshop on Description Logics (DL 2009)**; 2009
7. Li J: **Roadmap for Tool Support for Collaborative Ontology Engineering.** *Master of Science.* University of Victoria, Department of Computer Science; 2003.
8. Handel MJ: **What is Chat Doing in the Workplace?** In *Conference on Computer Supported Cooperative Work*; *Nov. 16-20; New Orleans, Louisiana*. 2002
9. Holsapple CW, Joshi KD: **A Collaborative Approach to Ontology Design.** *Comm ACM* 2002, **45:** 42-47.
10. Tempich C, Simperl E, Luczak M, Studer R, Pinto H-S: **Argumentation-Based Ontology Engineering.** *IEEE Intelligent Systems* 2007 **22:**52-59
11. Udell J (Ed.). ***Practical Internet Groupware***: O'Reilly; 1999.
12. Zhdanova AV: **Community-driven ontology construction in social networking portals.** *Web Intelligence and Agent Systems, IOS Press* 2006, **6:**93-121.