# Practical experiences in concurrent, collaborative ontology building 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.*

## Introduction

Engineering ontologies that are representative of a community consensus is of great interest to those working in bioinformatics and often requires close collaboration. 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]. Firstly, concurrent ontology editing, the ability for multiple edits to be made to the ontology at a single time and from different computers. Secondly, tracking annotations (called 'notes' in CP) associated with corresponding representational units (RUs). Thirdly, tracking annotations associated with actions of ontology change, such as deletions, axiom edits and annotation edits. Fourthly, a manageable mechanism for discussion threads and instant messaging for online editors that satisfy the need for communication between ontology developers that makes collaborative ontology building so useful.

The new Collaborative Protégé (CP) plugin [2] for the widely used open-source ontology editing tool Protégé 3, claims to support the above features. CP enables concurrent editing of a single OWL file. The tool 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, notes and discussions as instances of an integrated *Change and Annotation Ontology (ChAO)*, thereby providing an audit trail of edits and decision making. This tool, therefore appears an appropriate choice for an evaluation of collaborative ontology engineering and we present an initial investigation into its use.

## Materials and Methods

Thirteen members of the OntoGenesis Network came together at the European Bioinformatics Institute (EBI) for the 7th OntoGenesis Meeting (website: http://ontogenesis.ontonet.org/moin/NetworkMeeting 7). 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 enriched 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. 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).

Our methodology involved 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.

*Controlled additions:* Placement of selected device classes from the MSI term list into OBI. The appropriate metadata required by OBI were also added.

*'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 occurring to the other online editors.

*Controlled Communication:* Communication was restricted to specified channels during each editing session in order to evaluate CPs ability to foster communication, i.e. via notes, discussion threads and chat

Initially, development occurred in a single group but was then divided into two groups. *Ad hoc* additions were made, where editors were able to add and edit classes as they saw appropriate. Participants were then further divided in 4 pairs of 2, which then 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 by other pairs adding annotations. After more MSI terms were added by the whole group, first the chat was used to comment, annotate and discuss these additions. Then they were discussed by voice only and after that by chat and voice together. During the latter stages of this session, the *Agent Provocateur* user was deployed.

## Results

*Editing the ontology*

The complete editing metrics, together with tables, diagrams and deeper discussions, can be found in the supplementary material accessible from the 7th OntoGenesis Meeting website.

The OBI file grew 4.3% over the meeting course, whereby the increase in added defined classes (10.2%) was nearly double that of primitive classes (4.8%). Three new object properties were created during the meeting. These were used in a total of 68 new existential restrictions (9.7% increase). By inspection, increased chat indicates increased editing activity (see Table 2 of the supplementary information). The data also show different users working on different parts of the ontology and on different RU types. Apparent roles of users differed, e.g. 'moderators' creating tasks for others, which showed up in the metrics.

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. 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 just highlight areas that are currently worked on according to a colour scheme identifying the users, which 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.

Priority has to be the undoing of deleted classes, 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. A roll back function would aid in conflict resolution and would lead to a 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 it's 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. Getting notified on 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 on recent developments in their interest or responsibility-domain. 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 amount 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. 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 in a high level abstraction, rather than 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 Threads" 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. 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 is overkill. 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 requested by some users as well.

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.

Detailed statistics on numbers and kinds of annotations made during the sessions are available in a spreadsheet and diagrams in the supplementary material.

We positively note that in cases where the present (meta-) annotations are not sufficiently granular, the annotation types in CP’s underlying ChAO can be expanded with new annotation types that suit special projects needs and evaluation approaches.

Search and filtering of user annotations: It is possible to filter by author, annotation text, annotation type or by creation date, alone or in logical combinations. Own metadata schemes, e.g. certain obi annotation properties like `has_curation_status` or `definition_source`, can be queried for by the queries tab.

*Communication*

In the beginning, threads and notes were misused for chats and *vice versa*, the latter 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 nonetheless a warning had been issued, people used this obsolete property.

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 sociologic 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 been already realized at a certain time point. E.g. when people add new classes from a spreadsheet they should have a checklist that indicates which class has already been taken care of.

*Performance*

Overall, the performance of CP was very usable and much can be done with configuration to optimize it further. In large artifacts, expanding the full class hierarchy at once for the first time in one client can take its time (ca. 20 sec in our setup). Also opening a class with many direct subclasses for the first time will slow down and impair performance initially.

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.

## Conclusion

In this investigation, a realistic collaborative ontology building session was created using CP and its features were thoroughly tested. Areas where user requirements were not fulfilled have been highlighted. 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.

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 enhance annotation traceability of the tool further. In all, CP as it stands now is already usable as a collaborative tool that we can recommend. Our analysis provoked much feedback to the CP developers, and will be valuable for the CP version of P4, which is currently in preparation. Further use of CP in controlled settings will enable us to acquire further insights into the process of tool-based collaborative ontology building and such findings will be fed back to tool development in the future.

## Acknowledgements

*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 to Melanie Courtot, Alan Ruttenberg 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). He was supported by EBI's NET-project funds (www.ebi.ac.uk/net-projects) during some initial work leading to this paper. DS likes to thank his former and current employers Susanna Sansone and Stefan Schulz 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] Noy N, Chugh A, Alani H, The CKC Challenge: Exploring Tools for Collaborative Knowledge Construction. BMIR-2007; p. 1260.

[2] Tudorache T, Noy NF, Tu SW, Musen MA, Supporting collaborative ontology development in Protege, Seventh International Semantic Web Conference, 2008, Karlsruhe, Springer, Germany