The State of Multi-User Ontology Engineering

Julian Seidenberg and Alan Rector

Bio-Health Informatics Group
University of Manchester, United Kingdom
jms@cs.manchester.ac.uk, rector@cs.manchester.ac.uk

Abstract. Collaborative ontology engineering enables the creation of large complex ontologies. However, few projects successfully perform such multi-user ontology modeling. This paper gives an overview of ten different ontology engineering projects' infrastructure, architecture and workflows. It especially focuses on issues regarding collaborative ontology modeling. The survey leads on to a discussion of the relative advantages and disadvantages of asynchronous and synchronous modalities of multi-user editing. This discussion highlights issues, trends and problems in the field of multi-user ontology development.

1 Introduction

Semantic technologies give rise to new opportunities in many scientific disciplines where complex knowledge and processes need to be recorded. Organizations are increasingly taking advantage of semantic technologies for modeling this knowledge. Description logic based knowledge representation languages such a OWL make such modeling possible. Skilled ontology engineers can create ontological models of the real world and use tools to check their consistency, as well as infer information that is only implicitly contained in the model; they thereby can serendipitously discover new knowledge [10].

As these ontologies grow larger they expand beyond the capabilities of a single person. A team of multiple ontology engineers is required to build and maintain such large ontologies. However, there are only few modeling projects that manage to successfully perform multi-user ontology development. This paper discusses some of these projects and the design choices and trade-offs they faced. This paper also offers suggestions for improved editing tools and editing methodologies.

Our survey methodology involved conducting detailed interviews with members of ten different ontology engineering projects. Interviews were conducted either by phone or in person. The ontologists were asked to describe their organization and modeling process in detail. They were also asked to complete a questionnaire and encouraged to elaborate verbally on their answers.

2 Asynchronous vs. synchronous editing

The majority of the projects questioned in the user study perform only very basic multi-user ontology development. That is, they allow only one person to edit the ontology at any one time. Some projects edit ontologies asynchronously, while others synchronously work on a single shared knowledge base. A discussion of the relative advantages and disadvantages of these approaches follows.

2.1 Traditional concurrent editing

There are many different models for changing traditional structures like databases, computer programs and text documents. Ontologies differ from these traditional document types. Ontologies are used to represent knowledge using description logic. This allows the meaning of statements in an ontology to be expressed in such a way that a computer can understand and make inferences about them. Some traditional systems also capture semantics at a schematic level. These systems do not, however, usually allow frequent and/or dynamic modification of these schemata.

Specifically, ontologies written in the OWL description logic [6] can be classified by a description logic reasoner, which allows errors (logical contradictions) to be found, implicit knowledge to be made explicit and complex queries to be answered. However, with the additional power of these semantic tools come new possibilities for mistakes and conflicts. For example: in a multi-user scenario one person might define a VegetarianPizza as not having Meat and not having Egg as an ingredient while someone else creates an OmeletPizza as a subclass of VegetarianPizza as shown in Figure 1. The conflict between OmeletPizza and VegetarianPizza could not occur in a traditional database-like application, since such systems generally do not capture semantics.

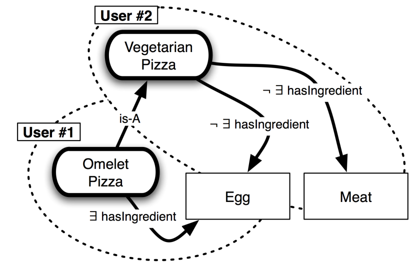

Fig. 1. Example of conflict that can only occur in a semantic multi-user editing scenario

3 Project descriptions

A total of ten different projects were evaluated in this user study. The projects were the following:

3.1 Project overviews

OpenGALEN [1] The GALEN project created a large ontology of human anatomy, pathophysiology, function, surgical procedures, pathology, diseases and drugs.

The ontology is divided into 2738 modules organized across three mono-hierarchies (trees) that conflate both compile-time dependencies and editorial groupings. Any number of such modules could be locked and checked-out by any of the twenty members of the multi-national development team. Some modellers would check out modules for many weeks.

Locks created some degree of safety between the modelers, as well as allowing individual modelers to develop from an infrequently changing baseline. However,

[1] `http://www.opengalen.org`

occasionally there were problems with two editors independently creating the same concept and/or referencing concepts that did not yet exist.

New releases of the ontology were published roughly every six months. Construction of a new release required all modelers to return and unlock all their locked modules. The complete ontology was then processed by a single curator who iteratively performed tasks such as correcting syntax errors, classifying the complete ontology, correcting any violations of asserted invariants and removing reported redundancy until the ontology classified without error. The episodic curation process typically took one to two weeks, in part because classification alone was a lengthy (2-3 hour) process. Any further modeling work done during this curation time needed to be repeated after the completion of the new release.

Ordnance Survey [2] The Ordnance Survey collects and maintains very detailed topographic data for Great Britain. It is building an ontology of relationships between topographic features. Of the three people in the organization who routinely edit ontologies, only one engineer is allowed to make changes to this ontology at any one time. Changes are discussed in face-to-face meetings or via email.

Anesthesia Patient Safety Foundation Data Dictionary Task Force [3] The APSF-DDTF is using a simplified cut-down version of the Protégé OWL ontology editor [4] to model an ontology of anesthesia. The ontology is largely based on a small section of SNOMED CT, but also captures some unique aspects of anesthesia that do not make sense in the context of SNOMED. It is edited in face-to-face meetings with a single scribe making all of the changes that a group of experts agree upon. At any other time members informally agree on who will take up the baton and edit the ontology.

Additionally, on rare occasions, some multi-user concurrent editing is done. Independent parts of the ontology are edited concurrently. The changes are then manually stitched together into a single ontology using a XML editing tool.

BioPax Consortium [4] The BioPax Consortium is a world-wide effort to develop a multi-level ontology of biological pathway information. The ontology file is emailed between members for editing. Only one person changes the file at any one time. Changes are discussed in conference calls.

U.S. National Cancer Institute [5] The National Cancer Institute in the United States is continually refining the NCI-Thesuarus, a large biomedical ontology of diseases, therapies and other topics related to cancer research.

The ontology is currently being maintained using a set of tools from the Apelon corporation. A group of 16 editors work on the ontology. These editors import a copy of the ontology into their own private individual database and make changes independent of each other. Every month all changes are sent to a central curator who integrates the changes and resolves any conflicts. The curator

[2] `http://www.ordnancesurvey.co.uk`
[3] `http://www.apsf.org`
[4] `http://www.biopax.org`
[5] `http://www.cancer.gov`

works for three to eight hours to produce a new updated baseline version of the knowledge base. This baseline is then re-imported by all the editors and they again start working independently.

The conflict resolution step is a bottleneck in the NCI process. While the curator is working to produce a new baseline all the other editors are idle. Editors may not make any changes during that time, since these would be lost with the release of the new baseline.

The NCI must balance the loss of productivity while a new baseline is created with the danger of two editors independently creating concepts of the same name, or worse: concepts with different names that were meant to express the same thing. Ideally, infrequently creating new baselines minimizes the time lost. However, frequently creating new baselines minimizes the possibility of problems due to overlapping work.

Another important issue for the NCI is the lack of integration between tools: the workflow, conflict resolution, editing and publishing tools are separate and must be manually integrated. For example: editors must manually email their change set to the curator for integration.

In order to address its modeling difficulties the NCI is planning to switch to a different editing paradigm. They want to host their ontology on a central database server to which individual clients connect using Protégé OWL. Any changes immediately show up for all connected editors. If two modelers start editing the same concept at the same time, one modeler's changes will be lost when the other commits.

Ontology for Biomedical Investigations [6] OBI is an international effort to collaboratively build an ontology for biological and medical investigations (formally known as the Functional Genomics Investigation Ontology (FuGO) project).

The project currently includes 13 specialized communities with a total of 70 developers collaborating across four different countries. Only one person may access the ontology at any one time. Major changes are made in face-to-face meetings every six months with a single scribe editing. Minor changes require approval from everyone in the community to make sure that no one else is already working on a change. Since the community is world-wide, time differences between countries result in a two day delay for even the most minor change (e.g. adding a comment).

The project uses Protégé OWL to edit an ontology stored in a subversion repository. Side projects are sometimes launched as branches from the main ontology. These must be manually merged into the main ontology once complete.

Imperial Cancer Research [7] Imperial Cancer Research at University College London is building an OWL ontology of diseases and treatments related to breast cancer. Existing ontologies were found to be too large and confusing to use. The ontology is being maintained by a single person.

[6] `http://obi.sourceforge.net`

[7] `http://www.cancerresearchuk.org`

Systematized Nomenclature of Medicine [8] The SNOMED RT (Reference Terminology) project was a large effort to create very large reference terminology of medicine. At its peak, the project employed thirty domain experts around the USA to concurrently construct the ontology. It was completed in the year 2001.

The SNOMED RT project endeavored to create a very high-quality knowledge base by using dual independent review. Sets of concepts are assigned and distributed out to modelers. Each concept is sent to at least two different modelers. These modelers create definitions for their assigned concepts and then send them back into a central server. An automated conflict resolution tool analyses the resulting definitions and, if the two modelers definitions don't match, initiates a conflict resolution procedure. The conflict resolution tool allows modelers to discuss and eventually reach a consensus on how to best model the term in question. 2 to 6% of concepts usually result in conflicts, though this is very much dependent on the nature of the subject area being modeled. Some areas produce no conflicts, while in others almost every concept results in a conflict [1].

SNOMED CT (Clinical Terms) is continuing the development of SNOMED merged with the CTV3 terminology (formally known as Read Codes) used in the United Kingdom. Contrary to the RT development, CT no longer uses dual independent review. The ontology is primarily edited synchronously in a central location (Chicago) with all modelers connecting to a central database. Additionally, three or four modelers/centers outside of the main location asynchronously edit branches of the terminology. These branches are integrated into the main ontology approximately every two weeks. While this integration is going on all asynchronous editing stops, but it is usually performed overnight, so little actual downtime occurs. Any conflicts that arise are assigned to and corrected by modelers in the central location. The integration results in a new baseline ontology which is distributed to the outside modelers for reviewed asynchronous editing.

Editing is performed using Apelon's Terminology Development Environment (TDE).

Clinical e-Science Framework [9] The CLEF project at the University of Manchester and University College London aims to integrate, organize and extract data from clinical information (e.g. medical records).

The two researchers at the University of Manchester have collaboratively constructed an ontology of medical concepts for use in the project. Only one person makes changes to the ontology at any one time. Changes are discussed face-to-face.

CancerGrid [10] CancerGrid is a consortium of ontology and software developers that is working on providing terminology servers to cancer research centers throughout the UK. They integrate various large ontologies (GO, SNOMED, NCI-Thesaurus) into each server and allow researchers to access, query, annotate and locally extend the unified knowledge corpus.

[8] `http://www.snomed.org`
[9] `http://www.clef-user.com`
[10] `http://www.cancergrid.org`

While CancerGrid is not directly developing an ontology of its own, it is very much involved in multi-user ontology engineering and provided us with requirements for future tools in light of its aims.

3.2 Project matrix

Table 1 shows modes of editing that various projects use. The most common workflow is to simply pass around a single file, so only one user edits the ontology at any one time. The table also details the projects' members' preferred mode of editing, assuming a choice of synchronous and asynchronous tools were available.

Project	asynchronous vs. synchronous	locking vs. optimistic non-locking
OpenGALEN	asynchronous	locking
Ordnance Survey	single-user (async preferred)	n/a
APSF-DDTF	single-user (sync preferred)	n/a
BioPax	single-user	n/a
NCI	asynchronous (apelon) synchronous (protégé)	non-locking non-locking
OBI	single-user (async preferred)	n/a
Imperial Cancer Research	single-user (async preferred)	n/a
SNOMED RT	asynchronous	non-locking
SNOMED CT	both synchronous and asynchronous	non-locking
CLEF	single-user (sync preferred)	n/a
CancerGrid	(asynchronous preferred)	n/a

Table 1. Modes of editing by project

4 Project types

4.1 Modes of editing

Multi-user ontology development can be divided into two basic modes of editing:

1. **Asynchronous editing**: multiple ontology engineers working on individual copies of a knowledge base, submitting their changes to a central server.
2. **Synchronous editing**: multiple ontology engineers symultaniously connected to a single knowledge base on a server with changes taking effect immediately.

Synchronous editing has the advantage that changes (and errors) are immediately visible to all parties involved. Another advantage is that there is only a single master version of the ontology at any one time. With asynchronous editing, there is always the potential of multiple branches [8] being created.

However, in a synchronous editing environment all parties must always be connected to the same server. Disconnected editing on trains and airplanes is not possible. This also prevents editors from going away and working in a sandbox environment to test a potentially harmful series of changes. In an asynchronous environment potentially dangerous changes can be independently developed, verified to be safe and then integrated with the rest of the ontology.

A mistake, conflict, or inconsistency can be disastrous in a synchronous environment. An error could result in the knowledge base becoming unusable for all connected users and editors. If the integrity of the knowledge base is critical then a synchronous environment is particularly dangerous. In contrast, in an asynchronous environment an error can be undone before it affects anyone else, thereby reducing the cost associated with fixing errors. Also, even if an error damages the complete knowledge base, its state can be selectively rolled back with relative ease; while, in a synchronous environment, rolling back the knowledge base to a previous state (or snapshot) will erase all the work done up to that point (just not the offending update), and stop everyone's progress while the roll-back is being executed.

The asynchronous workflow requires that changes be integrated into the complete knowledge base as editors submit them. This process is more complex than clients directly operating on the central ontology. Different lengths of check-outs create a degree of uncertainty about the state of the knowledge base. A modeler could at any time submit a significant change. Most projects using asynchronous editing therefore every so often collate all changes into a build/baseline/release version, rather than being in a state of continuous flux.

4.2 Methods of conflict prevention

Means of preventing conflicts can also be divided into two:

1. **Locking**: a section of the knowledge base is locked in order to prevent multiple people editing it at the same time.
2. **Optimistic non-locking**: locks are not used. The inevitable conflicts are fixed after they occur.

The advantage of locking is that it pro-actively prevents conflicts. However, editing becomes more difficult, especially as the number of locks increases and other users are prevented from accessing large portions of the knowledge base. Locking (in an asynchronous setting) also requires some means of modularizing or segmenting the ontology [7]. Optimistic non-locking methods removes barriers to editing, thereby encouraging widespread editing, but there is an increased effort required to fix errors.

Project management is more difficult using non-locking methods. The ability for anyone to edit anything at any time makes it difficult for modelers to be aware of each other's editing. Locks force users to edit their pre-declared or assigned areas.

In the absence of locks non-unique definition conflicts [1] can occur. That is, two modelers can create different definitions for the same concept in the knowledge base. This duplicate work is redundant. One editor's work will inevitably be lost. However, in certain circumstances such redundant work is desirable. For example: during the construction of SNOMED RT each concept was modeled by at least two people. Definitions were then compared and only accepted if all modelers agreed. This lead to a higher-quality knowledge base at the expense of duplicate work.

4.3 User preference

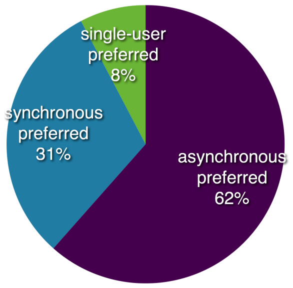

Fig. 2. Preferred mode of editing (n=13; some projects express multiple preferences)

The majority of ontology engineers questioned in our survey preferred an asynchronous approach to multi-user ontology engineering to a synchronous methodology (see Figure 2). Anecdotal evidence suggests that more technical ontology authors with backgrounds in software engineering tended to prefer the asynchronous model, while ontologists with life-science specializations, without backgrounds in computer science tended to prefer a synchronous model.

The more technical ontologists tended to be comfortable with asynchronous CVS-style systems and were aware of overlapping edit problems in a synchronous system. Less technical ontologists were unaware of potential problems and wanted a multi-user ontology editing to be as simple as possible. They regarded an asynchronous editing system as too complicated.

5 Project statistics

Statistics of the various project were compiled. These are as follows:

Ontology structure The ontologies under development have a median of 3000 classes (mean of 58380), with an estimated average of about 3 restrictions per class and a maximum number of about 25 restrictions on a single class. 60% of ontologies under development are single large ontology, with only 20% of projects using mutually importing ontology modules and another 20% creating multiple disconnected ontologies (see Figure 4).

Ontology criticality Most of the ontologies under development are not being used in critical systems within the organization. Errors mostly cause only minor inconvenience (see Figure 3).

Ontology language The majority of ontologies are being developed using Protégé OWL [4] utilizing the OWL ontology language, as shown in Figure 5.

Ontology maturity Half of the ontologies being developed by the surveyed projects are mature (GALEN, Ordnance Survey, Anastasia, NCI, SNOMED), which means they are well-established knowledge bases that are now being further refined. Other KBs are either in early or in the middle stages of development (as shown in Figure 6).

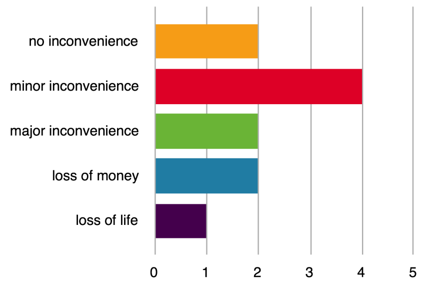

Fig. 3. Ontology criticality: consequence of errors in the ontology (x-axis: number of projects that describe themselves as belonging to the a specific category)

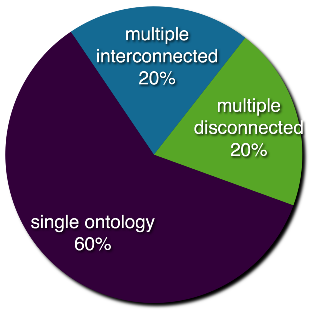

Fig. 4. Ontology structure (n=10)

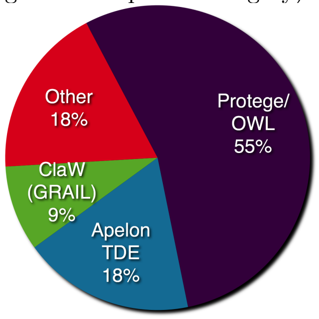

Fig. 5. Ontology development tools used (n=11; some projects use multiple tools)

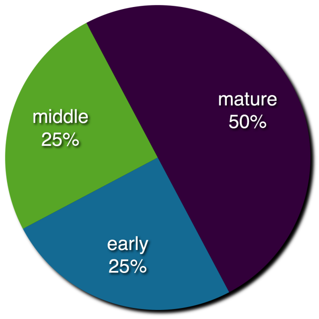

Fig. 6. Maturity of ontologies under development (n=10)

Organizations 50% of organizations surveyed operate internationally. 80% of organizations had multiple sites. 90% of organizations that perform multi-user ontology development have their modelers distributed in different physical locations.

Communication Email is the preferred means of communication for almost all organizations questioned. Only SNOMED and GALEN have dedicated workflow, conflict resolution and communication tools. Other communication technologies include: shared desktop clients, face-to-face meetings and the telephone. All of these are used to approximately the same degree, but none as universally as email.

6 Problems in multi-user editing

Multi-user ontology engineering is not very widely practiced. This section analyses why this is the case and what can be done to enable better concurrent ontology development.

A fixed questionnaire was used in our evaluation. Project representatives were encouraged to elaborate on their answers and/or suggest new alternative answers. However, beyond several adjustments resulting from a prototype questionnaire presented to two local projects (Clef and OpenGALEN), no additional categories were suggested. Interviews were conducted either by phone or in-person with one representative from each project.

6.1 General problems

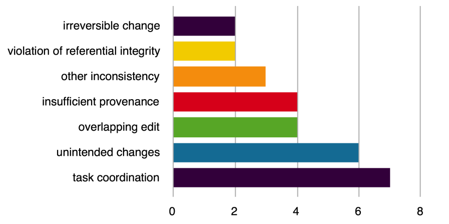

Fig. 7. Current problems in multi-user editing (x-axis: number of projects with a certain problem)

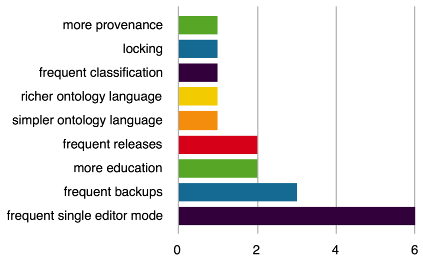

Fig. 8. Current solutions to enable multi-user editing (x-axis: number of projects using a certain solution)

Figure 7 shows the main problems in multi-user ontology editing. Most projects who's members we interviewed are distributed over multiple physical locations, many are even international (see section 5). This inevitably makes co-ordinating ontology modeling difficult. Keeping control of the editing process seems to be the main problem developers faced when attempting to do multi-user ontology development. Another major source of problems included unintended changes which occurred due to the concurrent nature of the editing process.

6.2 Current solutions

The most popular solution to concurrent editing problems is abandoning the idea altogether. As also mentioned in section 3.2, Figure 8 shows that most projects resort to passing their ontology between single individuals for single-user editing. This seems to be a stopgap measure to allow developers to at least make some progress. The general consensus is that ontology engineers would certainly prefer a true multi-user workflow, if it was available.

6.3 Barriers to better solutions

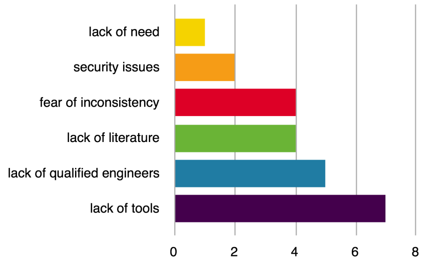

Fig. 9. What currently inhibits multi-user ontology development? (x-axis: number of projects seeing each inhibiting issue)

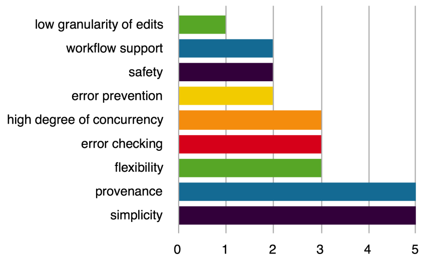

Fig. 10. Most important characteristics of a multi-user ontology editing system (x-axis: number of projects seeing each characteristic as desirable)

Figure 9 shows what is inhibiting organizations from performing their desired form of multi-user ontology editing. Having sufficient tool support is clearly critical for concurrent editing, with trained ontology engineers (and training material) also being very important.

6.4 Summary of problems

Concurrent editing of shared ontologies is both difficult and error prone. Good multi-user editing tools and quality training literature does not exist, so ontology engineers resort to makeshift solutions such single user editing.

7 Tool requirements and design considerations

This section analyses the requirements for tools to support multi-user ontology editing.

7.1 Tools overview

The most important tool in building ontologies is clearly the ontology editor. Popular ontology editors include: Protégé OWL [4], Top-Braid Composer [11], Swoop [3], Apelon Terminology Development Environment [12] and Altova SemanticWorks [13].

To our knowledge Protégé OWL offers some support for synchronous development and Apelon supports some asynchronous development. Still, both solutions could be significantly improved. For example, as far as we are aware, none of the above tools support any kind of locking.

7.2 Multi-user tool requirements

A multi-user tool requires a different set of features than a single-user application. Figure 10 shows which characteristics users in the questioned organizations considered most important for such a tool.

Unsurprisingly, simplicity was very important for the ontology engineers. They found current applications too complicated to use. However, these developers were equally concerned about provenance information. They wished that any multi-user tool automatically records as much provenance data as possible.

A low granularity of edits (the ability to select or edit a very small section of an ontology for a short time), was not seen as very important. This was probably due to ontologists not seeing the connection between this feature and the ability to achieve a high degree of concurrency, which was classed as an important feature. The lower to granularly of each edit, the more concurrent editors the tool can support.

Surprisingly, users also found error prevention, safety and workflow support to be relatively unimportant. These were perceived to add unnecessary complexity to the editing process. Users commented that they would rather have the ability to diagnose errors and rollback erroneous changes than be constrained by a system that imposes significant restrictions to ensure total safety when editing. Also, rather than be constrained by a fixed workflow, ontology engineers preferred a greater degree of flexibility.

[11] `http://www.topbraidcomposer.com`
[12] `http://www.apelon.com`
[13] `http://www.altova.com/features_owl.html`

7.3 Duration of edits

Asynchronous edit duration In an asynchronous editing scenario the length of time of a *check-out* is an important design consideration. The longer an ontology engineer independently works on a section of the ontology, the greater the chance for conflict. This is especially likely in an optimistic non-locking scenario. The greater the number of conflicts, the more time must be spent by a curator (potentially in single-user mode, see section 6.2) or by a conflict resolution system to negotiate a solution to the conflicts.

With that in mind, a longer duration of edit/check-outs should necessitate a system of stricter locks to prevent a greater number of conflicts. However, stricter locks also increase the barriers to editing (see section 4.2). This ability to carry out widespread edits is especially important with long check-out durations, since an ontology engineer might find that he or she needs to edit something that was recently locked by someone else and therefore is likely to remain locked for some time to come. So, in a long check-out scenario there is a conflict between conflict prevention and widespread editing. A system designer must decide if widespread editing or conflict prevention is the priority and choose to implement either non-locking or locking respectively.

This trade off is especially relevant, since we found that the majority of ontology engineers anticipated that, if working in an asynchronous ontology development system, their duration of edit would range between one to five days. This is illustrated by Figure 11.

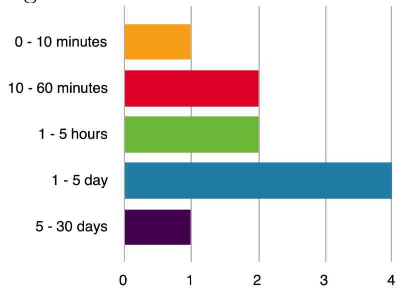

Fig. 11. Duration of check-out during asynchronous editing (x-axis: number of projects typically using check-outs of the specified duration)

Synchronous edit duration In a synchronous editing system multiple editors should be prevented from simultaneously changing the same axiom in an ontology. However, check-outs, as such, do not exist, so there is no need to consider duration of edits with relation to locking, beyond the simple case of overlapping edits.

7.4 Debugging considerations

When people build ontologies they inevitably make mistakes. If a statement in an ontology represents a logical contradiction, then a description logic classifier will highlight the unsatisfiable concepts. Debugging tools [9] [2] can then be used to discover the cause of the error. Error correction tools can then be used to find possible solutions [5]. In a multi-user environment error checking tools are even more important, since contradictions can be introduced at any time any by any of the ontologists working on the shared ontology. This is especially true if an ontology is being edited synchronously.

Figure 12 shows the frequency of certain categories of errors resulting from multi-user interactions, as well as how difficult these errors are to diagnose and fix, as reported by the interviewed ontology engineers.

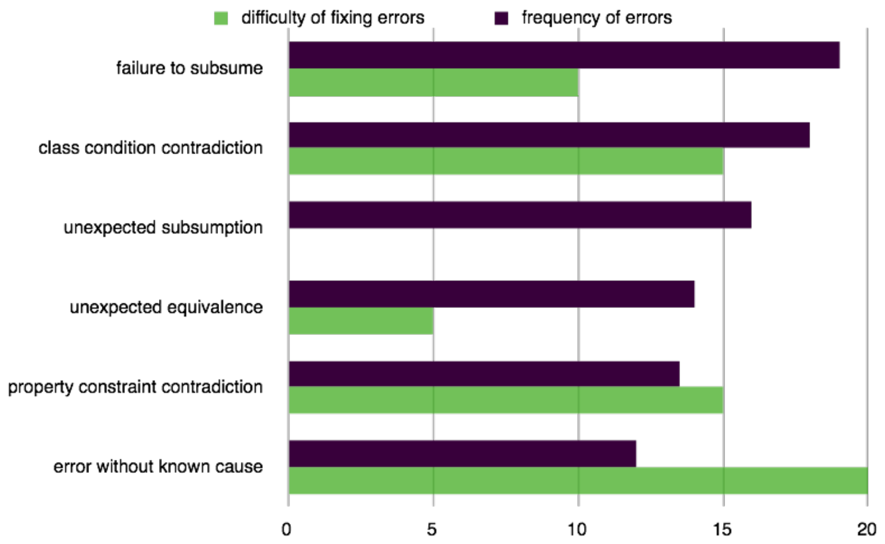

Fig. 12. Common errors in ontology modeling and effort involved in correcting them (x-axis: sum of all projects' 0-5 rating of each frequency/difficulty)

Errors without any known cause were the most difficult to fix, as might be expected, but these kinds of errors were also quite rare. Property statements contradictions were the second most difficult to fix. Debugging tools handle these kinds of errors quite well.

Failure to achieve the desired subsumption, while the most common error, was not as difficult to fix as the two aforementioned errors. So, correcting a contradiction was more difficult than achieving a desired subsumption. However, while ontology debugging tools help editors diagnose contradictions, they do not help with suggesting how to achieve desired subsumption relationships. That is: it would be good if the ontologist could indicate a desired subsumption relationship and the tool could generate a series of suggestions as to how to achieve this.

Interestingly, no one mentioned unexpected subsumption as even slightly difficult to fix, even though it was the third most encountered error. This seems to indicate that unexpected subsumption is less an error and more a natural part of ontology development. Simply because a new subsumption relation is unexpected does not mean that it that it is necessarily undesireable, quite the opposite, in fact.

8 Summary

In this paper we have explored the findings from our survey of ten ontology development projects with relation to multi-user ontology development. We found that most organizations find multi-user ontology development very difficult, primarily because of a distinct lack easy-to-use tools. Because of this most projects end up resorting to a pass-the-pen mode of editing where only one user may edit the ontology at any one time.

An asynchronous mode of editing using locking seems appropriate for a majority of organizations, based on their requirements, modeling styles and preferences. This style of editing can also aid task coordination, which is the single greatest problem reported during multi-user editing. Other modes of editing are also applicable for more specialized projects with specific requirements.

Development tools need to be further improved beyond their present state. The survey suggests that debugging and refactoring tools would greatly enhance the multi-user development process, but their existence is relatively unknown. Additionally, a use case for a new tool was found: such a tool, given a desired subsumption relation, would suggest changes that result in the manifestation of this relation.

All these factors point to an overarching need for a multi-user ontology development methodology. That is: a clear guide for how to perform large-scale ontology engineering.

References

1. K. E. Campbell, S. P. Cohn, C. G. Chute, E. H. Shortliffe, , and G. Rennels. Scalable methodologies for distributed development of logic-based convergent medical terminology. In *Methods Inf Med*, 1998.
2. A. Kalyanpur, B. Parsia, E. Sirin, and B. Cuenca-Grau. Repairing unsatisfiable concepts in OWL ontologies. In *European Semantic Web Conference*, 2006.
3. A. Kalyanpur, B. Parsia, E. Sirin, B. Cuenca-Grau, and J. Hendler. Swoop - a web ontology editing browser. *Journal of Web Semantics*, 4(1), 2005.
4. H. Knublauch, R. W. Fergerson, N. Noy, and M. A. Musen. The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. In *Third International Semantic Web Conference (ISWC)*, 2004.
5. J. Lam, J. Z. Pan, D. Sleeman, and W. Vasconcelos. A fine-grained approach to resolving unsatisfiable ontologies. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI-2006)*, 2006.
6. D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview, February 2004. W3C Recommendation.
7. J. Seidenberg and A. Rector. Web ontology segmentation: Analysis, classification and use. In *15th International World Wide Web Conference*, May 2006.
8. M. Voelkel and T. Groza. Semversion: an rdf-based ontology versioning system. In *Proceedings of the IADIS International Conference*, 2006.
9. H. Wang, M. Horridge, A. Rector, N. Drummond, and J. Seidenberg. Debugging OWL-DL Ontologies: A Heuristic Approach. In *Proceeding of the 4th International Semantic Web Conference*, 2005.
10. K. Wolstencroft, A. Brass, I. Horrocks, P. Lord, , U. Sattler, R. Stevens, and D. Turi. A little semantic web goes a long way in biology. In *4th International Semantic Web Conference (ISWC)*, 2005.