Developing ontologies in OWL: An observational study[1]

Martin Dzbor, Enrico Motta, Carlos Buil, Jose Gomez, Olaf Görlitz, Holger Lewen

KMi, The Open University, Milton Keynes, UK (m.dzbor, e.motta@open.ac.uk)
iSOCO, Madrid, Spain (cbuil, jmgomez@isoco.com)
ISWeb group, University of Koblenz-Landau, Germany (goerlitz@uni-koblenz.de)
Institute AIFB, University of Karlsruhe, Germany (hle@aifb.uni-karlsruhe.de)

1. Introduction: Evaluating ontology engineering

As is known in the human-computer interaction (HCI) domain, interactions involve the user, the technology, and the ways they work together. We expand these notions to human-ontology interaction with the aim to investigate how users interact with the networked ontologies in a realistic ontology lifecycle. In this paper, we describe a user study that we have carried out in order to improve our understanding of the level of support provided by current ontology engineering tools in the context envisaged by the NeOn project. That is, in a scenario when ontology engineers are developing complex ontologies by reuse, i.e., by integrating existing semantic resources.

Some work on evaluating tools for ontology engineering has been done in the past. In [2] authors conclude that the tools available in the time of their study (cca 1999) were little more than research prototypes with significant problems in their user interfaces. These included too many options for visualizing ontologies, which tended to confuse the user and hinder navigation. Moreover, the systems' feedback was found to be poor, which meant a steep learning curve for non-expert users. Finally, most tools provided little support for raising the level of abstraction in the modelling process and expected the user to be proficient in low-level formalisms.

Work described in [8] evaluated Protégé in several tasks, from the perspective of a power user. The authors found the system intuitive for expert knowledge engineers, as long as operations were triggered by them (e.g. knowledge re-arrangement). However, difficulties arose when assistance from the tool was expected; e.g. in inference or consistency checks. Weak performance was also noted in language interoperability. The survey reported in [3] also noted issues with tool support for operations on ontologies beyond mere editing (e.g. integration or re-use). In particular, the authors emphasised the limited 'intelligence' of current tools – e.g. no possibility to re-use previously used processes in current design. Tools expected the user to drive the interaction, with the tool imposing constraints rather than adapting itself to users' needs.

Finally, some researchers [11] found that visualization support in Protégé and its customization models are too complex and do not reflect users' models of what they would normally want to see. Others observed users having difficulties with description logic based formalisms in general [5]. Again, tools expected detailed knowledge of intricate language and logic details, and this often led to modelling errors.

To summarize, existing empirical work highlighted several problems with ontology engineering tools. However, at the beginning of the NeOn project we felt that there

[1] All authors and this study were supported by NeOn project (see http://www.neon-project.org)

was a need to conduct a novel study, as none of the studies mentioned above provided the kind of data we wanted to obtain as a baseline to inform the development of the next generation ontology engineering tools envisaged by NeOn. Specifically, the studies did not satisfactorily address the following key concerns of our project:

- **Emphasis on 'normal users'.** As ontologies become an established technology, it makes less sense to focus only on highly skilled knowledge engineers. There are so many organizations currently developing ontologies that it seems safe to assert that indeed most ontologies are currently built by people with no formal training in knowledge representation and ontology engineering. Therefore, it is essential to conduct studies which focus on 'normal users', i.e., people with some knowledge of ontologies, but who cannot be classified as 'power users'.
- **Emphasis on ontology reuse.** NeOn adopts the view that ontologies will be *networked*, *dynamically changing*, *shared* by many applications and strongly dependent on the *context* in which they were developed or are used. In such scenario it would be too expensive to develop ontologies 'from scratch', and the re-use of existing, possibly imperfect, ontologies becomes the key engineering task. Thus, it makes sense to study the broad re-use task for OWL ontologies, rather than focusing only on a narrow activity (e.g. ontology visualization or consistency checking).
- **Formal definition of ontology engineering tasks.** Studies reported earlier focused on generic tool functionalities, rather than specifically assessing performance on concrete ontology engineering tasks. This creates two problems: (i) the results are tool-centric, i.e., it is difficult to go beyond a specific tool and draw generic lessons on how people do ontology engineering tasks; (ii) by assessing the performance of our users on concrete tasks using OWL ontologies, we acquire robust, benchmark-like data, which (for example) can be used as a baseline to assess the support provided by other tools (including those we plan to develop in NeOn).

For these reasons we decided to conduct our own user study, addressing the three criteria described above. We describe the methodology in detail in the next section.

2. Observational user study

We conducted an *observational study* rather than an experiment to capture user needs and gaps in the tool support, rather than merely compare different tools. As mentioned earlier, NeOn is concerned with several facets of networked ontologies, and many of these facets are currently supported to a very limited extent. This lack of tools and techniques makes it difficult to assess the actual user performance in any of these tasks. However, it enables us to acquire generic requirements and insights on a broader ontology engineering *task* or *process*.

By definition [4], ontology is a shared artefact integrating views of different parties. One form of integration used in this study was *temporal*, where an agent re-used previously agreed ontologies, perhaps from different domains. All studied ontologies are publicly available, all are results of principled engineering processes and knowledge acquisition, and they all model domains comprehensible to a 'normal user'. Table 1 shows some statistical information on the studied OWL ontologies.

Table 1. Features of the ontologies used: Cl(asses), **Pr**(operties), **Re**(strictions)

Ontology	Cl	Pr	Re	Notes
Copyright	85	49	128	Mostly cardinality & value type restrictions, some properties untyped [http://rhizomik.net/2006/01/copyrightontology.owl]
AKT Support	14	15	n/a	All properties fully typed, no axioms [http://www.aktors.org/ontology/support]
AKT Portal	162	122	130	10 classes defined by equivalence/enumeration, most properties untyped [http://www.aktors.org/ontology/portal]

The *Copyright* ontology was set as a base to be adapted by re-using and integrating terms from the other two ontologies. Two environments were used – Protégé from Stanford University and TopBraid Composer from TopQuandrant – these satisfied the initial requirements from ontologies (e.g. on OWL fragment or visualization features)

We worked with 28 participants from 4 institutions (both academic and industrial). Participants were mixed in terms of different experience levels with designing ontologies and with different tools. Each person worked individually, but was *facilitated* by a member of the study team. Participants were expected to have knowledge of basic OWL (e.g. sub-classing or restrictions), while not necessarily being 'power users'. They were recorded with screen capture software Camtasia, and at the end they filled in a questionnaire about their experiences of different aspects of the development.

2.1 Tasks given to users

Participants were given three tasks considering different ways of integrating ontologies into a network. Task 1 was the simplest and most precisely set, and Task 3 the most complicated and requiring users to break the overall goal into operational steps.

In Task 1, participants were told that the *Copyright* ontology did not formalize temporal aspects, and had to be augmented with the relevant definitions from other ontologies (e.g. *AKT Support*). The objective was to review the three given ontologies, locate the relevant classes (i.e. *CreationProcess* and *Temporal-Thing*), import ontologies as needed, and assert that *CreationProcess* $\sqsubseteq$ *Temporal-Thing*.

Task 2 was motivated by pointing to a western-centric notion of any right being associated only with a person, which excluded collective rights. Participants were asked to review concept *copyright:Person*, and replace its use with deeper conceptualizations from the *AKT Portal* and *AKT Support* ontologies. In principle, the task asked people to express two types of restrictions on property ranges:

- **simple**: e.g. for concept *Economic-Rights* introduce *rangeOf* (*agent* , *Legal-Agent*);
- **composite**: e.g. state that *rangeOf* (*recipient* , (*Generic-Agent* $\sqcap$ ($\neg$ *Geo-Political*)))

Task 3 asked people to re-define concept *copyright:Collective* so that formal statements could match an informal description. Participants were told to make amendments in the base – *Copyright* ontology, rather than to the other two. We expected they would first create new local sub-classes for the concept *copyright:Collective*, and then make them equivalent to the actual AKT classes, i.e.:

- Create new concept *copyright:myOrganization* $\sqsubseteq$ *copyright:Collective*
- Make it equivalent to an existing concept *myOrganization* $\equiv$ *akt:Organization*
- Prove that the tool 'knows' about the intended meaning (by e.g. using an inference tool to classify the classes as *akt:Organization* $\sqsubseteq$ *copyright:Collective*)

The second half of Task 3 comprised a definition of a new property (e.g. *copyright:hasMember*) with appropriate domain and range, together with its restriction for class *copyright:Collective*, so that a collective is defined as containing min. 2 persons.

2.2 Evaluation methodology

We opted for a *formative evaluation* [9] to inform design of new OWL engineering tools in the context of NeOn. Two constraints were observed: (i) gathered data shall not be tool-specific (it was not our objective to prove which one tool was best); and (ii) while generic tool usability was considered important, measures were expected not to be solely usability-centric. In terms of what was analyzed, we selected the following levels of analysis [6]: (i) user's satisfaction with a tool, (ii) effectiveness of a tool in achieving goals, and (iii) behavioural efficiency. In our study, these categories took the form of questions exploring *usability*, *effectiveness*, and *efficiency* categories, to which we added a generic *functional assessment* category.

Our questionnaire reflected issues that typically appear in the literature correlated with enhancing or reducing effectiveness, efficiency, usability, or user satisfaction [10] (36 questions). The remaining 17 questions inquired about various functional aspects considered relevant to the NeOn vision; incl. ontology re-use, visualization, contextualization, mapping, reasoning, etc.

The questionnaire included both *open* and *closed (evaluative)* questions. The former asked for opinions; the latter used a Likert scale ranging from very useful (+1) to very poor (–1). For each such question we calculated a mean, which is listed below (sign 'minus' denoting a negative attitude). We also expressed frequencies and counts – largely in the context of open, qualitative items and observations. Positively and negatively stated questionnaire items were interspersed to avoid the tendency of people to agree with statements rather than disagree [1]. Nevertheless, this tendency towards agreeing appeared during analysis; e.g. in an overall neutral mark to the tool's design quality, despite complaining consistently about its features.

3. Selected findings

This section summarizes key findings from the study. For each category of measures we give a general summary of observations across the whole population, followed by commenting on differences (if any) between two common denominators of user performance in knowledge-intensive tasks – the choice of and the expertise with the tool.

3.1 Effectiveness-related observations

Here we explore how effectively the tools were in carrying out the tasks. We look at measures such as complexity of getting acquainted with the tool, support for repetitive activities, and overall tool behaviour. Table 2 summarizes a few general observations, and Table 3 compares several features where differences were observed.

As shown in the tables, the help from tools to get users acquainted with their GUI-s was rated slightly negative. Participants were not convinced by the tools' capability to reduce the complexity by providing its functions simply and effectively. A stronger perception of shortcomings emerged for frequently repeated operations (e.g. repeated definition modifications). Taking user comments in account, they had to, for example, repeat a search operation not for the sake of finding a class, but as *an impromptu 'spell check'*. In this context the overall impression of both tools was rather negative.

Table 2. Selection of a few general observations across population

Measure/question	–1	0	+1	Total	Mean
process of getting around the tool and understand it	23%	61%	16%	31	–0.065
support for frequently repeated operations	43%	50%	7%	31	–0.367
overall behaviour of the tool	7%	90%	3%	31	–0.032
effectiveness of dealing with task 2 (e.g. difficulties)	27%	54%	19%	31	–0.161
effectiveness of dealing with task 3 (e.g. difficulties)	11%	67%	22%	31	+0.194
help from the facilitator	3%	45%	52%	31	+0.483

Table 3. Comparison of attitudes between tools and expertise groups (TB: TopBraid, Pr: Protégé, Be: less experienced, Ex: expert); significance threshold: χ^2=5.99 at p=0.05

Measure	Type	Outcome	χ^2	Sign.
overall behaviour of the tool	tools	TB (0.0) vs. Pr (-0.143)	3.14	no
subjective complexity of Task 1	tools	TB (-0.70) vs. Pr (-0.33)	2.17	no
subjective complexity of Task 3	tools	TB (-0.10) vs. Pr (+0.33)	4.09	no
process of getting around the tool and understand it	experience	Be (+0.12) vs. Ex (-0.27)	3.02	no
role of tool in reducing complexity of Task 2	experience	Be (-0.44) vs. Ex (+0.13)	9.71	yes
role of tool in reducing complexity of Task 3	experience	Be (-0.06) vs. Ex (+0.47)	5.63	no

In terms of tool assessment, TopBraid users rated the tool neutral, whereas Protégé users showed more negative attitudes. Despite TopBraid being unfamiliar, people found it easier to understand than Protégé; this correlates with a slightly simpler interface of TopBraid and the 'multi-dimensionality' of its GUI (e.g. enabling work with multiple ontologies or clearer structuring of inference). Another possible explanation is that TopBraid is based on a standard Eclipse environment, which is shared with other development tools (e.g. Java), and therefore, tends to be familiar to the users.

Nonetheless, both tools were judged as generally unhelpful in supporting frequent operations, with similar problems: confusion with the import function, non-standard icons, dialogs or mouse interactions. A minor variance was observed in Tasks 2 and 3, where less experienced participants suffered more from the lack of tool support. Otherwise, both groups judged the overall tool behaviour as neutral to slightly negative. More extreme reactions came from the experts regarding support for frequent operations and understanding the tool's functions. Qualitatively, experts were suggesting improvements in using standard features (e.g. delete or move functions), and also in interaction modalities. An extreme reaction quoted from one user was: "*Too much mouse interaction. […] Feels like programming with the mouse*".

3.2 Efficiency-related observations

Here we look at such measures as how efficient people felt in different tasks, how they were assisted by the help system or tool tips, how the tools helped to navigate the ontologies or how easy it was to follow the formalisms used in definitions. Table 4 shows general observations, and Table 5 compares features with some differences.

Table 4. Selection of a few general observations across population

Measure/question	–1	0	+1	Total	Mean
providing sufficient information about ontologies	32%	55%	13%	29	–0.172
support provided by documentation, help	60%	40%	0%	16	–0.500
usefulness of the tool tips, hints, …	50%	46%	4%	27	–0.423
subjective time taken for task 2	25%	55%	20%	31	–0.065
subjective time taken for task 3	6%	56%	38%	31	+0.300

Table 5. Comparison of attitudes between tools and expertise groups (TB: TopBraid, Pr: Protégé, Be: less experienced, Ex: expert); significance threshold: χ^2=5.99 at p=0.05

Measure	Type	Outcome	χ^2	Sign.
subjective speed of completing task 1	tools	TB (-0.80) vs. Pr (-0.33)	2.94	no
help with handling ontology dependencies	tools	TB (0.0) vs. Pr (-0.37)	7.65	yes
useful visualization & ontology navigation facilities	tools	TB (-0.33) vs. Pr (-0.63)	6.00	yes
handling ontology syntax / abstract syntax	tools	TB (+0.40) vs. Pr (-0.07)	2.33	no
ease/speed of carrying out mappings	experience	Le (-0.21) vs. Ex (+0.27)	9.75	yes
level of visualization and navigation support	experience	Le (-0.69) vs. Ex (-0.40)	2.40	no
ontology representation languages, abstract syntax, etc.	experience	Le (-0.22) vs. Ex (+0.23)	3.64	no
management of versions for engineered ontologies	experience	Le (+1.0) vs. Ex (–0.50)	3.37	no

The efficiency of the two tools was approximately the same. There was a slightly faster performance of TopBraid users compared to Protégé, but only in Task 1. This might be due to a slightly simpler import function in TopBraid, which was the only major challenge of Task 1. When asked about efficient handling of ontology dependencies and navigating through them, Protégé users thought they were significantly less efficient. Many users were not happy with the abstract syntax of the axiom formulae, which was not helped by the inability to edit more complex restrictions in the same windows and wizards as the simple ones.

One qualitative feature in both tools concerns the depth of an operation in the user interface. Subjectively, 32% participants felt they had an explicit problem with finding an operation in a menu or workspace. The main 'offenders' were the *import* function (expected to be in File → Import… menu option) and the *in-ontology search* (which was different from the search dialog from Edit → Find… menu option).

Expertise seemed to have minimal effect on the assessment of the efficiency dimension. Both groups concurred that while a lot of information was available about concepts, this was not very useful, and the GUI often seemed cluttered. They missed a clearer access to 'hidden' functions such as defining equivalence or importing ontology. Non-experts saw themselves inefficient due to lack of visualization and navigation support, and also due to the notation of abstract DL-like formalism. Experts were at ease with the formats; non-experts considered support for this aspect not very good.

The overwhelming demand was for complying with common and established metaphors of user interaction. A quote from one participant sums this potential source contributing to inefficiency: "*More standard compliance and consistency. The search works differently … usual keyboard commands … don't always work…*"

3.3 Design and user experience related observations

Two key aspects were evaluated with respect to user experiences – (i) *usability* of the tool (which included accessibility and usefulness), and (ii) more general *satisfaction* with the tool. The latter included comments regarding user interface intuitiveness, acceptability, customization, and so on.

Table 6. Selection of a few general observations across population

Measure/question	–1	0	+1	Total	Mean
usability of tool's help system	60%	40%	0%	16	–0.500
usefulness of the tooltips, hints, …	50%	46%	4%	27	–0.423
support for customization of the tool, its GUI or functionality	48%	44%	8%	25	–0.400
usefulness of handling ontology dependencies	31%	66%	3%	27	–0.259
visualization of imports, constraints & dependencies	58%	39%	3%	28	–0.536
support for [partial] ontology import	62%	14%	4%	29	–0.739
useful tool interventions in establishing mapping	48%	52%	0%	26	–0.480

Table 7. Comparison of attitudes between tools and expertise groups (TB: TopBraid, Pr: Protégé, Be: less experienced, Ex: expert); significance threshold: χ^2=5.99 at p=0.05

Measure	Type	Outcome	χ^2	Sign.
level of overall satisfaction with the tools	tools	TB (+0.10) vs. Pr (-0.19)	2.67	no
overall satisfaction with tool's environment	tools	TB (+0.10) vs. Pr (-0.24)	3.14	no
support for handling dependencies among ontologies	tools	TB (0.0) vs. Pr (-0.37)	7.65	yes
level of visualization and navigation support	tools	TB (-0.33) vs. Pr (-0.63)	6.00	yes
ease of carrying out concept mapping	tools	TB (+0.50) vs. Pr (+0.10)	5.85	no
usefulness of the tooltips, hints, …	experience	Be (-0.25) vs. Ex (-0.57)	2.45	no
availability of tool customization, its GUI or functionality	experience	Be (-0.64) vs. Ex (-0.21)	7.90	yes
effort to get acquainted with the tool	experience	Be (-0.27) vs. Ex (+0.12)	3.02	no
overall satisfaction with tool functionality	experience	Be (-0.33) vs. Ex (0.0)	3.10	no
support for reasoning and inferences	experience	Be (0.0) vs. Ex (+0.07)	3.19	no
support for multiple ontology representation formats	experience	Be (-0.22) vs. Ex (+0.23)	3.64	no

As Table 6 shows, responses in this category are generally negative; participants considered the existing support as "very low" or "not very good". Almost invariably, they were dissatisfied with the role of documentation, help system, tool tips and various other tool-initiated hints. Support for tool customization – i.e. either its user interface or functionality – was also inadequate. A common justification of the low scores was (among others) the lack of opportunity to automate some actions, lack of support for keyboard-centric interaction, lack of support for more visual interactions. As can be seen from these examples, the reasons were quite diverse, and to some extent depended on the user's preferred style.

One emerging trend on the tools' usability was that too many actions and options were available at any given point during the integration tasks. On the one hand, this refers to the amount of information displayed and the number of window segments needed to accommodate it. An example of this type of usability shortcoming is the (permanent) presence of all properties on screen. On the other hand, while constant presence can be accepted, it was seen as too rigid – e.g. no filtering of only the properties related to a concept was possible. In fact 32% claimed that unclear indication of inheritance and selection was a major issue, and further 14% reported being unable to find all uses of a term (e.g., property or concept label) in a particular ontology. Other comments related to usability are summarized below:

- *unclear error messages and hints* (e.g. red boundary around an incorrect axiom was mostly missed);
- *proprietary user interface conventions* (e.g. icons for browsing looked differently, search icon was not obvious, some menu labels were misleading);
- *lack of intuitiveness* (e.g. finding an operation in the menu, flagging the concept in the ontology so that it does not disappear, full- vs. random-text search capabilities);
- *inconsistent editing* & amending of terms (e.g. while "subClassOf" was visible at the top level of the editor, "equivalentTo" was hidden)

As shown in Table 7, a significant difference of opinion was in the overall satisfaction with the tools, their design and intuitiveness, where it was more likely that people complained about Protégé than TopBraid. In this context, we can see that people tended to be more positive in the abstract than in the specific – responses to specific queries were negative (between –0.500 and -0.100), yet overall experiences oscillate between –0.111 and +0.100. As we mentioned, the overall satisfaction with the TopBraid environment was more positive (some possible reasons were discussed above).

One case where experience weighed strongly on less experienced users is the tool intuitiveness. Probably the key contributing factors were the aforementioned non-standard icons, lack of standard keyboard shortcuts, ambiguous operation labels, and an overall depth of key operations in the tool. Less experienced users also had issues with basic features – e.g. namespaces and their acronyms, or ontology definition formalisms. The issue with formalisms is partly due to the inability of the tools to move from an OWL- and DL-based syntax to alternative views, which might be easier in specific circumstances (such as modification of ranges in Task 2). Experienced users missed functionalities such as version management – here less experienced users were probably not clear in how versioning might actually work in this particular case.

Discussion

Technology (such as OWL), no matter how good it is, does not guarantee that the application for its development would support users in the right tasks or that the user needs in performing tasks are taken on board. At a certain stage, each successful tool must balance the technology with user experience and functional features [7]. This paper explored *some persevering issues* with OWL engineering tools that reduce the appeal and adoption of otherwise successful (OWL) technology by the practitioners.

As shown above, although the tools made a great progress since the evaluations reported in section 1, issues with user interaction remain remarkably resilient. The effort was spent to make the formalisms more expressive and robust, yet they are not any easier to use, unless one is proficient in the low-level languages and frameworks (incl. DL in general and OWL's DL syntax in particular). Existing tools provide little help with the user-centric tasks – a classic example is visualization: There are many visualization techniques; most of them are variations of the same, low-level metaphor of a graph. And they are often too generic to be useful in the *users'* problems (e.g. seeing ontology dependencies or term occurrences in an ontology).

Table 8 highlights a few gaps between what is provided by the current tools and what people see as useful for framing problems in a more user-centric way. Some 'wishes' (white rows) already exist (e.g. Prompt for version comparison), but perhaps our findings may further improve design of the existing OWL engineering tools.

Table 8. Summary of gaps vs. support for partial fixes

Measure or remedy	**Avg. mark (–) current, (+) future**
Existing support for ontology re-use	–0.097 (not very good)
Support for partial re-use of ontologies	–0.739 (very poor)
→ flag chunks of ontologies or concept worked with	+0.519 (*would be* very useful)
→ hide selected (irrelevant?) parts of ontologies	+0.357 (*would be* useful)
Existing support for mappings, esp. with contextual boundaries	–0.065 (not very good)
Management and assistance with any mappings	–0.480 (not very good / poor)
→ query ontology for items (instead search/browse)	+0.433 (*would be* useful)
→ compose testing queries to try out consequences of mappings	+0.045 (*would be* possibly useful)
Existing support for versioning, parallel versions/alternatives	–0.200 (not very good)
Existing visualizing capabilities & their adaptation	–0.536 (very poor)
→ mechanism to propagate changes between alternative versions	+0.519 (*would be* very useful)
→ compare/visualize different interpretations/versions	+0.700 (*would be* very useful)
→ opportunity to perform operations in graphical/textual mode	+0.414 (*would be* useful)
→ visualize also on the level of ontologies (not just concepts)	+0.357 (*would be* useful)

Table 9. Observations of issues with OWL engineering and user interaction

Observation	**Frequency**	**% affected**	**Examples**
Syntactic check (brackets, logic) → user not alerted or not noticing	21x	64.3%	Buttons/icons after axioms misleading; Single/double clicks to select, edit, etc.
Testing (inference, meaning, interpretation,…)	26x	64.3%	Which inference is the right one?; How to check the intended meaning(s)?
Translate/compose logical operation (e.g. equivalence, inheritance)	37x	60.7%	How to start complex axiom?; Stepwise definition? …
Dialogs, buttons,… (confusion, inconsistency)	43x	89.1%	Buttons/icons after axioms misleading; Single/double clicks to select, edit, etc.
Searching for the class (partial text search on labels)	25x	64.3%	Labels starts with X different from label contains X; namespaces in search?
Functionality unclear (drag&drop, error indication, alphabetic view)	26x	60.7%	Am I in the edit mode?; Where is it alerting me about error?;

For instance, frequently used operations and their correlations provide us with opportunities to improve the support. In our case, the support was given by facilitators, but clearly, the support for the frequent actions is likely to affect the experiences with OWL engineering. The most frequent steps in OWL development are the actual coding of definitions and import of ontologies (unsurprisingly), but, surprisingly, also search (71% users), re-conceptualization of restrictions and editing of logical expressions (both 54%), and locating terms in ontologies (46%). Compare these operations with the situations requiring assistance from facilitators (in Table 9).

Correlations were observed between, e.g., incorrect logical conceptualization and confusion caused by ambiguous labels or dialogs. Other correlations were between problems with importing an ontology and absence or semantic ambiguity of appropriate widgets in the workspace, and between difficulties with definitions and the failure of tools to alert users about automatic syntactic checks (e.g. on brackets). The translation of a conceptual model of a restriction into DL-style formalism was a separate issue: 70% were observed to stumble during such definitions. From our data, we suggest considering *multiple ways* for defining and editing axioms (to a limited extent this partly exists in Protégé). Any way, DL may be good for reasoning, but it is by no means the preferred "medium for thinking" (even among ontology designers).

Another issue is the gap between the language of users and language of tools; a high number of users was surprised by syntactically incorrect statements. In 64.3% sessions at least one issue due to syntax (e.g. of complex restrictions) was observed. Because of these minor issues they had to be alerted to by facilitator, people tended to doubt results of other operations (e.g. search or classification) if these differed from what they expected. Lack of trust is problematic because it puts the tool solely in the role of a plain editor, which further reduces tool's initiative. In an attempt to restore 'user trust', some tools (e.g. SWOOP) move towards trying to justify their results [5].

The extensive use of features in the tools is also an issue increasing complexity of user interaction. Both tested tools showed most of *possibly* relevant information on screen *at all times*. There was little possibility to filter or customize this interaction. The granularity at which tools are customizable is set fairly high. For instance, one can add new visualization tabs into Protégé or use different (DIG-compliant) reasoning tool, but one cannot *modify or filter the components of user interaction*.

Clearly, there is some way to go to provide the level of support needed by 'normal' users engineering OWL ontologies. Our analysis highlighted some shortcomings, esp. the flexibility and adaptability of user interfaces and lifting the formal abstractions. With this study, we obtained a benchmark, which we plan to use to assess the support provided by our own future tools in 18-24 months. Obviously, we intend to include other OWL engineering tools (e.g. SWOOP or OntoStudio) to make the study robust.

References

[1] Colman, A.M., *A Dictionary of Psychology*. 2001, Oxford: Oxford University Press. 864p

[2] Duineveld, A.J., Stoter, R., *et al.*, *WonderTools? A comparative study of ontological engineering tools.* Intl. Journal of Human-Computer Studies, 2000. **52**(6): p.1111-1133.

[3] Fensel, D. and Gomez-Perez, A., *A survey on ontology tools*. 2002, OntoWeb Project.

[4] Gruber, T.R., *Towards principles for the design of ontologies used for knowledge sharing.* Intl. Journal of Human-Computer Studies, 1993. **43**(5/6): p.907-928.

[5] Kalyanpur, A., Parsia, B., Sirin, E., *et al.*, *Debugging Unsatisfiable Classes in OWL Ontologies.* Journal of Web Semantics, 2005. **3**(4).

[6] Kirkpatrick, D.L., *Evaluating Training Programs: the Four Levels*. 1994, San Francisco: Berrett-Koehler Publishers. 289p.

[7] Norman, D., *The Invisible Computer*. 1998, Cambridge, MA: MIT Press.

[8] Pinto, S., Peralta, N., and Mamede, N.J. *Using Protégé-2000 in Reuse Processes*. In *Evaluation of ontology-based tools (EON)*. 2002. p.15-25.

[9] Scriven, M., *Beyond Formative and Summative Evaluation*, In *Evaluation and Education: A Quarter Century*, McLaughlin, M.W. and Phillips, D.C., Eds. 1991, University of Chicago Press: Chicago. p.19-64.

[10] Shneiderman, B. and Plaisant, C., *Designing the User Interface: Strategies for effective human-computer interaction*. 4 ed. 2004: Addison-Wesley. 672p.

[11] Storey, M.A., Lintern, R., Ernst, N.A., *et al. Visualization and Protégé*. In *7th International Protégé Conference*. 2004. Maryland, US.