Contents lists available at ScienceDirect

Int. J. Human-Computer Studies

journal homepage: www.elsevier.com/locate/ijhcs

Overcoming the pitfalls of ontology authoring: Strategies and implications for tool design[☆]

Markel Vigo[*], Samantha Bail, Caroline Jay, Robert Stevens

School of Computer Science, University of Manchester, Manchester, UK

ARTICLE INFO

Article history:
Received 11 February 2014
Received in revised form
17 June 2014
Accepted 21 July 2014
Communicated by N.F. Noy.
Available online 30 July 2014

Keywords:
Ontologies
Semantic web
Authoring tools

ABSTRACT

The process of authoring ontologies appears to be fragmented across several tools and workarounds, and there exists no well accepted framework for common authoring tasks such as exploring ontologies, comparing versions, debugging, and testing. This lack of an adequate and seamless tool chain potentially hinders the broad uptake of ontologies, especially OWL, as a knowledge representation formalism. We start to address this situation by presenting insights from an interview-based study with 15 ontology experts. We uncover the tensions that may emerge between ontology authors including antagonistic ontology building styles (definition-driven vs. manually crafted hierarchies). We identify the problems reported by the ontology authors and the strategies they employ to solve them. These data are mapped to a set of key design recommendations, which should inform and guide future efforts for improving ontology authoring tool support, thus opening up ontology authoring to a new generation of users. We discuss future research avenues in light of these results.

© 2014 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/3.0/).

1. Introduction

The use of ontologies for capturing the knowledge of a domain of interest has grown significantly, with areas such as bioinformatics and healthcare modelling and sharing information in ontologies of varying size and complexity. One of the central ideas of the Semantic Web is that of sharing, linking, and reusing data from multiple sources; as a consequence, the availability of (high quality) semantically described data sources and thus the uptake of Semantic Web technologies are important, not only to the success of the Semantic Web, but also to applications in which rich domain descriptions can play a role.

Considering the inherent complexity of a knowledge representation formalism such as OWL, convincing domain experts, and thus, potential ontology authors, of the usefulness and benefits of using ontologies is one of the major barriers to broader ontology adoption (Hepp, 2007). The following excerpt, which was extracted from the interviews that inform the study presented here, is a clear example of the current situation:

> “A domain expert has to be totally convinced that ontologies are the right way of modelling knowledge in a domain, and then has to consistently work for a period of time in order to be self-sufficient. In the initial stages, when they start doing the modelling, you need a knowledge engineer to hold their hand […] the moment the knowledge engineer disappears they will not carry on with it because it's much easier to get in databases or Excel than to do all this.” (Participant 13)

There are several factors that may prevent the uptake of ontologies including the cognitive complexity of ontology languages such as OWL (Horridge et al., 2011), the step learning curve of these languages (Warren et al., 2014) or the required minimum expertise to model very specific domains (Randall et al., 2011). One of the key factors, in addition to the ones mentioned, that contribute towards the acceptance of a new technology is the availability of adequate tool support that allows both novice and expert users to create high quality artefacts that meet their functional requirements. While ontology authoring tools have seen many improvements over recent years, most notably demonstrated by the popularity of Protégé as an integrated ontology development tool (Cardoso, 2007), we still lack a good understanding of the effectiveness of existing tools, or what kind of tool support ontology authors need to successfully create, explore, reuse, refactor, and debug ontologies. The quote above is indicative of the complexity of the ontology authoring process, and how typical ontology engineering tools are unsuccessful in making the process of authoring ontologies easier and more accessible; a shortfall which affects not only the uptake of ontology by new users, but also the quality of artefacts created by existing ontology engineers.

[☆] This paper has been recommended for acceptance by N.F. Noy.
[*] Corresponding author.
E-mail address: markel.vigo@manchester.ac.uk (M. Vigo).

http://dx.doi.org/10.1016/j.ijhcs.2014.07.005
1071-5819/© 2014 The Authors. Published by Elsevier Ltd. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/3.0/).

In order to gain insights about the problems encountered in the authoring process we carry out an interview study with 15 ontology authors. We deliberately target ontology experts for two reasons: firstly, their understanding of problems is more comprehensive and richer; and secondly, they employ sophisticated authoring strategies that may be indicative of latent problems of tools. With the help of a qualitative analysis software tool, transcribed interviews were thematically analysed in an open coding fashion. A codebook of themes was agreed among the authors and an analysis of reliability was conducted to discard any source of bias (see the study in Section 3). The identified problems and strategies may not be representative of the whole spectrum of ontology authors though: novices may struggle with additional problems to the ones we identify; also, the participants were mostly drawn from life sciences and thus, some of the problems might be domain specific. We argue that emergent design recommendations will also entail benefits for novices and across disciplines as there are commonalities—of a different magnitude—between the problems faced by different types of users. For instance, the cognitive complexity of OWL is challenging for all users even if it is a more severe barrier for non-experts.

We believe that, in order to improve the experience of ontology authors, it does not suffice to only analyse their interaction with ontology engineering tools; instead, we need a more *holistic* view of the ontology authoring process and lifecycle. In Section 4.1 we first focus on the different approaches that exist to authoring ontologies, including the different ontology building styles (what we here call ‘definition-driven’ ontology development vs. manually crafted hierarchies). We then need to gain a clear picture of the problems ontology authors encounter when developing ontologies, common tasks with which they struggle, and the tools and strategies they employ to overcome such problems (see Section 4.2). As a result, the contributions of this study are the following:

- We raise awareness about the latent tensions about ontology building styles in Section 5.1. While we do not take a stance in whether having a common understanding and strategy is right or not, having this awareness may help to remove frictions in teams.
- By mapping the problems and strategies to design insights, we suggest how the next generation of authoring tools could cater for the needs of broader audiences (see Section 5.2).
- In Section 5.3 we enumerate a set of operable design recommendations that emerge from the acquired insights.
- We evaluate state-of-the-art authoring tools against the design recommendations (see Section 5.4).

While some of our findings confirm existing assumptions about the ontology engineering process, such as the lack of tools for helpful exploration and visualisation of ontologies, others are more surprising. Amongst others, both preventing errors and debugging erroneous ontologies are still considered to be challenging tasks when constructing ontologies, while support for judging the quality of ontologies in the context of their functional requirements, in the spirit of unit tests in traditional software engineering, is not widespread. Furthermore, as reasoning is used frequently to check the correctness of ontologies, on-the-fly reasoning capabilities seem to become increasingly relevant.

2. Related work

Over the years, the usability of the tools of the trade has not been a major concern in the ontology engineering community. Pioneering work exists in the realm of knowledge representation systems, where it is indicated that explanation of inferences, adequate reporting of errors, and performance are key to the usability and success of such tools (McGuinness and Patel-Schneider, 1998).

One other systematic approach to the hitherto usability analyses of ontology authoring tools included an evaluation framework containing 26 dimensions to compare six ontology authoring tools (Duineveld et al., 2000). A set of dimensions was used to heuristically assess usability qualities through a number of statements such as “Evaluate the clarity of the interface” or “Is there a good overview of the ontology?”, whereas another set of questions articulated the features supported by tools in terms of “Is it possible to use multiple inheritance?” or “Does the tool check new data for consistency?”. The main conclusion of this work was that authoring tools of that time (circa 2000) were unusable for domain and ontology experts due to difficulties learning to use the tools and their poor usability.[1]

When it comes to ontology authoring tools, literature surveys are the most comprehensive and replicable method by which tools are classified and assessed against evaluation frameworks. Katifori et al. (2007) provide a detailed analysis discussing how authoring tools implement ontology visualisations such as zoom features or 3D navigation techniques. They suggest that visualisation should not be used as an isolated technique, but complemented with efficient search and navigation mechanisms. Also, they suggest that future ontology visualisation tools should be able to remove clutter and lead ontology authors to their goals. Usability, interoperability, and portability are some of the dimensions included in a framework to classify semantic authoring tools of textual content (Khalili and Auer, 2013). In this study, faceted browsing, faceted viewing, single point of entry interface, and inline editing were identified as the features that increase the usability of semantic content authoring tools. It is remarkable that out of the 175 papers analysed in the survey only 2 involve end-users in order to evaluate the user interface. In both cases enquiry methods were employed a posteriori in order to provide evidence about user acceptability of the evaluated tools. We do not know whether this small extent of user involvement can be generalised to general purpose ontology authoring tools. However, it can be considered an indicator of low involvement of users in the development process.

In a landscape where user tests are scarce, there are some notable exceptions of ontology authoring tool evaluation with users. In addition to evaluating tools against predefined criteria (stability, availability, and extendibility, to name a few), Lambrix et al. (2003) presented a study in which 8 non-expert users carried out a number of basic tasks including ontology loading, entity addition, modification, and removal. After completing their tasks, users were given questionnaires in order to quantify the relevance of tools to accomplish tasks, tool efficiency, user attitude towards tools, and learnability aspects. They concluded that none of the tools stood up for all tasks, but each of them had their weaknesses and strengths.[2]

A later user study from 2006 involved 28 participants with basic OWL knowledge carrying out three tasks with TopBraid Composer[3] and Protégé (Dzbor et al., 2006). Tasks included creating classes and properties, as well as adding subsumption, equivalence, and range axioms. Afterwards, participants were given questionnaires to measure the effectiveness, efficiency, and user experience with the tools. Dzbor et al. (2006) conclude that

[1] Tools evaluated were Ontolingua, WebOnto, Protégé Win, OntoSaurus, ODE and KADS22.

[2] Again, this research dates back to 2003, so it is unknown how this analysis applies to the current situation. Evaluated tools were Protégé 2000, Chimaera, DAG-Edit and OilEd.

[3] http://www.topbraidcomposer.com/

tools are not easy to use and criticise that the abundant ontology visualisation techniques are mostly based on the graph metaphor. Flagging parts of the ontology that have already been developed, hiding irrelevant sections of the ontology, and mechanisms to compare different versions are the visualisation features most demanded by participants.

One specific area of ontology research where we can find a series of user studies is that of explanation and debugging. These studies investigate the suitability of different approaches to presenting explanations in the form of *justifications* for (potentially erroneous) entailments to ontology engineers. They mostly focus on measuring the time and success rates of the debugging tools, but do not offer any further insights into how users interact with the explanations (Kalyanpur et al., 2005; Lam, 2007). The first major study on the cognitive complexity of justifications was carried out by Horridge et al. (2011). The study involved 14 participants who were presented with a set of items consisting of a set of axioms alongside a single axiom (the candidate entailment), and were asked to answer whether the candidate entailment followed logically from the axiom set or not. A high error rate indicated an item that was 'difficult' to understand for the test participants. The study also used eyetracking data in order to investigate where ontology authors fixate when given OWL justifications as stimuli. It was inferred that longer fixations were indicative of the complexity of the axiomatic constructs in justifications. Nguyen et al. (2012) presented an investigation of the difficulty users have with understanding various sub-patterns that occur in OWL justifications. This study used a 'mechanical turk' type web platform (Buhrmester et al., 2011) where test participants were presented with a natural language reasoning problem which was directly based on one of 51 justification sub-patterns that were automatically identified in a large corpus of justifications. The correct answers ranged from 100% to only 2% of the test participants, indicating that there exist reasoning patterns in OWL which are extremely difficult to understand; however, the study did not offer any explanation as to *why* these constructs are difficult for users.

In a study that analysed the cognitive complexity of OWL axioms (Warren et al., 2014), 12 researchers were given 21 questions containing a set of OWL statements and an inference. The participants had to tell whether or not the inference was right. These axioms, which were represented in a simplified version of Manchester OWL syntax were drawn from a survey that indicated the most frequently used OWL constructs (Warren, 2013). Participants performed worse ($\leq 50\%$ of correct responses) in those statements that contained (i) the complementation of the *and* operation (i.e., *not* (*A and B*)), (ii) the fact that transitivity is not inherited and (iii) functional characteristics.

While controlled laboratory experiments are necessary to isolate variables to ensure the internal validity of the research, they are often criticised due to the lack of ecological validity and intrusiveness. A major criticism of the studies we analysed is the small number of participants involved (fewer than eight users in some cases), which does not guarantee enough statistical power to strongly support the claims made, especially if questionnaires are used for data collection. A weakness of questionnaires is that preconceived questions get predefined responses and participants are unable to express additional thoughts that would have been interesting for researchers. Furthermore, studies based on predefined tasks, such as repairing an erroneous entailment, do not offer any insights into what makes this task easy or hard for a user. In order to counterbalance the problems of laboratory studies and questionnaires, laboratory experiments can be complemented with remote log analysis of ontology authoring tools which do not impose restrictions on users and allow the involvement of a higher number of participants. However, due to the quality of the collected data these studies may be less meaningful; see Wang et al. (2013) as an example.

Previous work on ontology authoring dynamics addresses the socio-technical issues of distributed collaborative authoring, highlighting that reaching consensus on the use, purpose and scope of a given ontology may generate tensions among the ontology authors (Randall et al., 2011). In contrast to collaborative ontology building, we explore axiom addition, which is carried out individually. This study complements the one by Randall et al. (2011) in that our approach directly asks ontology authors open-ended questions in order to ascertain the problems they encounter when performing authoring tasks individually.

3. Study setup

In this section, we briefly describe the study setup, including the participants, the interview method, and the interview analysis. In particular, we discuss the distinction between the three different types of study participants we identified in the interviews.

3.1. Method

Fifteen geographically distributed ontology authors were recruited for the study through convenience/snowball sampling: the participants who were initially recruited provided the contact details of other potential participants that fitted in the profile sought by the authors of the study. Participants were selected by neither ontology language nor authoring tool they used, but because their experience in authoring ontologies. Participants worked in a wide variety of disciplines: computer science, genetics, anatomy, archaeology, biology, and the food industry. The semi-structured interviews, which were conducted either face-to-face or via video-conferencing, followed this script:

- Can you describe the authoring tasks you perform?
- How do you use tools to support you in these tasks?
- What sort of problems do you encounter?

On average, the interviews took 41:31 min (*SD*=15:20 min). The interviews were recorded and transcribed.

3.2. Participants

We interviewed 15 ontology authors (4 male and 11 female) with an average age of 37 years (*SD*=7). Thirteen of these were based in the UK and two were based in the US. Seven were computer scientists, six were bionformaticians, and the remaining two had a background in other sciences. We targeted expert ontology authors which is supported by the fact that all of them had more than four years of experience working with ontologies. Participants were rewarded with a £10 book store gift voucher.

We classified the participants' use of ontologies into three groups (five participants in each): ontology *researchers*, *curators*, and *developers*. Ontology researchers are typically computer scientists who investigate the properties of ontologies or build reasoning engines; they frequently build test ontologies (*toy ontologies*) for the purpose of testing their developed systems (these ontologies are 'toy' in the sense that the ontologies are likely not to be used in a service setting, but nevertheless can be non-trivial in size and complexity). Curators are individuals with a deep knowledge about a domain who maintain ontologies that are often large and complex (at the scale of hundreds of thousands of classes and axioms). They receive requests from ontology users through issue tracking systems or from a consortia to add terms, update entities, or add branches into existing ontologies. Ontology developers are

Table 1
Participant profiles; codes for background column, CS: Computer Scientist, BIO: Bioinformatics, SCI: other sciences.

ID	Role	Background	Tools	Ontology domain
P1	Researcher	CS	Protégé, semantic wikis, OWL API	Test ontologies, life sciences
P2	Researcher	CS	Protégé, OWL API	Test ontologies, multimedia, medical
P3	Researcher	SCI	Protégé, OWL API	Test ontologies, software
P4	Researcher	CS	Protégé, OWL API	Test ontologies
P5	Researcher	CS	Protégé	Test ontologies, law
P6	Curator	BIO	Protégé, BioPortal	Life sciences
P7	Developer	SCI	Protégé, OWL API, text editor	Medical
P8	Developer	CS	Protégé, OBO-Edit, text editor	Life sciences, metadata, software
P9	Developer	CS	Protégé	Life sciences
P10	Curator	BIO	ArrayExpress, BioPortal, Corona, Zooma	Life sciences
P11	Developer	BIO	Protégé, BioPortal, CMAP, Ontobee, Ontodog, Ontofox	Life sciences, medical, metadata, software
P12	Curator	BIO	Protégé, AmiGO, OBO-Edit, Ontogene, QuickGO, OLS	Life sciences
P13	Developer	CS	Protégé, OWL API, TopBraid	Life sciences, archaeology, industry
P14	Curator	BIO	Protégé, OBO-Edit	Life sciences
P15	Curator	BIO	Protégé, OBO-Edit, text editor	Life sciences

computer scientists that work very closely with domain experts in order to model and validate a specific domain. The resulting ontologies are used by other applications, such as search engines or web applications.

Table 1 shows a summary of the participant profiles including the tools they use for ontology authoring and the domains of the ontologies with which they work. Most participants had mainly worked with ontologies belonging to the biomedical domain. Those involved in the medical domain reported to have worked with the OpenGalen and SNOMED CT ontologies, whereas a broad range of ontologies were dealt with by those working in the life sciences domain, including ChEBI, the Gene Ontology (GO), EDAM, and EFO.

The main selection criteria to recruit participants was experience in authoring ontologies. We did not seek users of particular tools or ontology languages in order to not to focus on a specific tool. The tools in Table 1 illustrate those reported by the participants. It can be observed that all users but one use Protégé for their tasks, making it the prevailing tool in this setting. However, this does not mean that Protégé is the tool that ontology authors mainly use, as many participants reported they only use it for specific tasks such as for visualising the ontology, transforming non-OWL ontologies into OWL and using the reasoner. This may indicate that, as we discuss later in Section 4.2, users turn to Protégé to carry out the tasks that cannot do or find difficult to accomplish with the tools they use by default.

3.3. Analysis

The transcriptions of the interviews were uploaded to the qualitative data analysis software, Dedoose 4.5.[4] Transcripts were thematically analysed in an open coding fashion following established analysis methods (Braun and Clarke, 2006): (1) familiarisation with data, (2) generating the initial codes, (3) searching for themes, and (4) iteratively reviewing themes. The generated codebook was agreed between the study's authors. Then, an independent coder (one of the study's authors who was not involved in the study design or codebook generation) was given the codebook and transcripts in order to establish coding reliability.[5] The inter-coder reliability analysis for the transcribed interviews yielded Cohen's $\kappa = 0.61$, which is considered a substantial agreement (Landis and Koch, 1977).

[4] http://www.dedoose.com/
[5] Interview transcripts and the codebooks are available online: http://wel-data.cs.manchester.ac.uk/studies/4.

4. Results

The emergent themes identified from this coding fall into two categories: on the one hand, users exhibit different perspectives and attitudes when authoring ontologies, which we consider to be 'tensions' (Section 4.1) between the fundamental approaches to modelling ontologies. On the other hand, we identify the common problems ontology authors encounter in the ontology development process, and the strategies they employ to overcome such difficulties (see Section 4.2), which extends our initial approach (Vigo et al., 2014a) by going deeper into the ontological aspects of problems and strategies. A discussion and interpretation of the study results will follow in Section 5.

4.1. Tensions revealed: definition-driven vs manually crafted hierarchies

OWL, coupled with an automated reasoner, allows the development of potentially highly complex ontologies. A *definition* of a class is an axiom (or a set of axioms) which may describe necessary and sufficient properties of the class. Given a set of definitions using equivalence axioms, an OWL reasoner will construct the (implicit) class hierarchy, while also checking that the ontology as a whole is coherent and consistent given the statements made. We call such an approach '*definition-driven*' based on the notion of '*formal definitions*' (Cimino, 1998). This definition-driven approach includes, but is not limited to, the technique known as 'normalisation' (Rector, 2003), where the different axes of classification are strictly separated into trees and the polyhierarchy reconstructed via defined classes using relationships between those trees; definition-driven approaches have in common the use of equivalence axioms and automated reasoning to help maintain the hierarchy, but the use of equivalence axioms is not itself a sufficient condition for being ontology normalisation. A definition-driven approach stands in contrast to an approach we call '*manually crafted hierarchies*' where the polyhierarchy of the ontology is largely asserted manually.

"*An ontology is a set of axioms, end of story*." (P2). This view, which is mostly held by computer scientists, entails that these authors create a set of axioms and let the reasoner build the hierarchical relationships between nodes: "*I try to keep the asserted classification to a minimum, I want the reasoner, the logic to determine the classification as much as possible*." (P15). Authors focus first on the relationships and then on the classes: "*We think about relationships before we actually talk about the actual terms, the terms come after*." (P6). This definition-driven approach is seen to have several advantages: "*Definition-oriented development is*

Table 2
Typical problems encountered and reported strategies to overcome these problems at each task. We report occurrences of strategies across ontology researchers (R), curators (C) and ontology developers (D).

Task	Problem tackled	Strategy	Total	R	C	D
Knowledge acquisition and modelling	Lack of domain expertise	Collaborate with domain experts	*6*	0	3	3
		Use external sources: books, journals, Web	*3*	1	1	1
Exploring the ontology	Lack of contextual awareness	Explore the class hierarchy	*6*	4	1	1
	Lack of familiarity and information overload due to large size	Navigate the class hierarchy	*3*	0	3	0
	Lack of support for searching across external ontologies in the preferred environment	Search on ontology repositories	*6*	1	4	1
	Lack of support for visualising ontologies in the preferred environment	Turn to Protégé to visualise the ontology	*8*	3	2	3
Ontology building	Inefficient to add a high number of entities	Use of alternative population methods: use of the OWL API and spreadsheets	*9*	3	3	3
	Avoid the creation of entities that already exist	Matching, mapping & merging using search features of ontology repositories	*8*	2	3	3
Reasoning	Resource-consuming	Externalise reasoning	*3*	0	2	0
		Remove complex entities	*3*	1	1	1
		Use low expressivity languages	*2*	1	1	0
	Lack of support for reasoning in the preferred environment	Turn to Protégé to run the reasoner	*5*	0	3	2
Debugging	Lack of support for debugging in the preferred environment	Injecting test axioms	*3*	2	1	0
		Frequent reasoning	*8*	2	4	2
	Poor feedback	Remove key errors	*2*	0	0	2
		Trial & error	*3*	1	1	1
		Reason about the problem	*2*	2	0	0
Evaluation	Lack of domain expertise	Consult experts	*4*	1	2	1
	Lack of evaluation support in the preferred environment	Competency questions	*3*	1	1	1

better for maintenance because the hierarchies are not any more subject to human decision […] *you don't have to decide upfront where you want to locate your thing*." (P5). Alternatively, authors assert as much as possible of the polyhierarchies. Some authors, especially those that have to release frequent updates stick to this 'manually constructed hierarchies' style in order to avoid reasoning: "*We have to assert everything because we don't reason on the fly*." (P12).

Even if some authors acknowledge the superiority of definition-driven development, there are some perceived limitations to embracing this approach. On the one hand it is more complex: "*It needs really advanced experience to do it*." (P1); on the other hand, asserting everything is considered by some to come more naturally: "*I don't minimise assertions, that's not in the human nature*." (P5). Limitations in tool support for definition-driven ontologies is seen as another problem, driving authors to build hierarchies by hand. "*this interface* [*Protégé*] *makes us make use of the hierarchy feature* [*as*] *the main structuring block for classes*." (P5).

Among those who develop ontologies with manually crafted hierarchies, different building methods are reported. Some exhibit a top-down approach where authors first create the top-level classes of the hierarchy and then add subclasses to them. In the bottom-up approach, the leaf nodes are created first and then classified according to some criteria. These criteria are used to create superclasses of the leaf nodes. This process is repeated until the root node is reached. A number of authors report methods that are in between the top-down and bottom-up approaches.

4.2. Problems and the strategies employed

We enumerate the problems encountered by participants when carrying out tasks of knowledge acquisition and modeling, exploration of the ontology, ontology building, reasoning, debugging and evaluation. These tasks are the themes that emerged from our analysis of the datasets in which participants relate their experiences as ontology authors. It is not incidental that the tasks correspond to different stages of the ontology authoring lifecycle. Considering that these tasks are a small subset of the processes and activities contemplated by existing methodologies (Suárez-Figueroa et al., 2012), it is not our intention to discuss them in the context of the lifecycle, but to focus on the problems encountered and the strategies employed in order to overcome these problems. The fact that some other important issues for ontology authoring are not mentioned (e.g., metadata authoring) indicates that our participants did not find relevant problems when conducting these activities, but by no means denies their importance to those ontology authors.

Table 2 comprises the strategies employed to tackle the problems encountered at each task. The last four columns of Table 2 indicate the number of authors that report their strategies to address the problems found at each task.[6]

4.2.1. Knowledge acquisition and modelling

Participants indicate that when modelling ontologies, it is preferable to be familiar with the domain, although most of the time this knowledge does not necessarily have to be very deep. Ontology authors tend to learn about the domain from different sources: "*I pull together lots of different papers, text books, classic books from the past 50 years and then try to synthesise that*." (P15). Sometimes that is not enough and if the topic is controversial, discussions with domain experts need to be held. P15, who is a bioinformatician, after exhausting all the options consulted domain experts: "*Are there 12 or 11* [*entities of the domain*] *here, and where do they connect? I have people looking down at the microscope and telling me*." (P15). A graphical representation of the ontology helps domain experts in particular: "*If there was a tool to help in visual modelling, that would help. Visual tools are the best for domain experts*." (P13). As expected, those who model ontologies that are used beyond testing in an ontology research context (curators and developers) are the individuals that turn to domain experts.

[6] Regarding the numbers in Table 2, note that the answers are non-exclusive, i.e. some study participants may have given multiple differing responses.

4.2.2. Exploring the ontology

Exploring the class hierarchy. These strategies are related to the loss of contextual awareness and to the problems of comprehending the current state of the tool and the ontology: "*You just lose the overview. If you have a small ontology I know every single axiom I put in there, I can remember it, whereas with big ontologies I wouldn't be able to remember every single existential restriction.*" (P4). Contextual awareness is mostly lost when invoking commands that modify the ontology beyond the authors control:

- After running the reasoner: "*People add properties to classes all the time and I run the reasoner. There is something that's gonna come that didn't come across before.*" (P15).
- After undoing the latest modifications: "*Undoing is very scary* […] *you press undo and maybe you already navigated away from the view* […] *and you don't know what you actually undo so I never use undo, never!*" (P5).

Authors get an overview from looking at the top element of the hierarchy, especially when they are not familiar with the domain (i.e., ontology researchers): "*The first thing would be to start from the top level of genes and just investigate the hierarchy and try to learn about it.*" (P1).

Navigating the class hierarchy. Navigation problems describe the difficulties encountered when navigating and orientating within the ontology because of its large size (in the case of curators) or due to the lack of familiarity with it: "*We have classes and instances about clinical statements; moving from instances to classes is not so good because instances appear as a plain list* […] *As the number increases it is more difficult to track back which instance belongs to which class.*" (P7). If the author is familiar with the domain and the ontology (i.e., curators), finding one's way through the ontology is done by navigating the class hierarchy, which is shaped as an expandable tree in typical ontology authoring tools (see Fig. 1): "*If you edit an ontology full-time you get very familiar with it* […] *all of us can drill down the ontology and find it quite easily.*" (P12). These strategies differ from exploration strategies in that, while the ontology authors are trying to understand the state of the ontology through exploration, in a navigation task they are looking for a specific entity or axiom related to an entity.

Search on ontology repositories. Search problems are caused by unsatisfactory searching capabilities of tools, or because tools cannot search across *multiple* ontologies. The latter is crucial as before adding new classes to ontologies, authors check the existence of similar terms in other ontologies in order to reuse them. "*I forgot what the primary label was; then it is very difficult because you cannot search on synonyms or other metadata.*" (P14). Domain-specific information retrieval engines allow authors to obtain information about classes and finding whether a given class exists in another ontology. "*I would look at the ontology directly in BioPortal to try and figure out what they mean.*" (P10). Search features are mostly used by curators, who normally have to make minor updates in very large ontologies.

Turn to Protégé to visualise the ontology. Those participants that do not stick to one authoring environment do turn to Protégé to visualise the ontology for different reasons:

- To have an overview and visualise the effect of actions. This was mainly reported in order to get visual feedback of the consequences of executing programmes developed through the OWL API or scripts that modify an ontology. "*The time when you use Protégé is when you run these scripts and then you want to have a look and say, OK, it still looks sensible.*" (P6). This strategy was also exhibited to become familiar with ontologies with which authors were not familiar by having a look at the class hierarchy: "*When I download a new ontology from the Web and I want to look around, I always use Protégé as it is easier to open up the ontology and have a look at the class hierarchy*" (P4).
- To show ontologies to domain experts. "*We show the ontology to the clinicians. If you give them the whole ontology that's not really useful for them.*" (P11). This strategy was reported only by developers, who tend to work closely with domain experts.

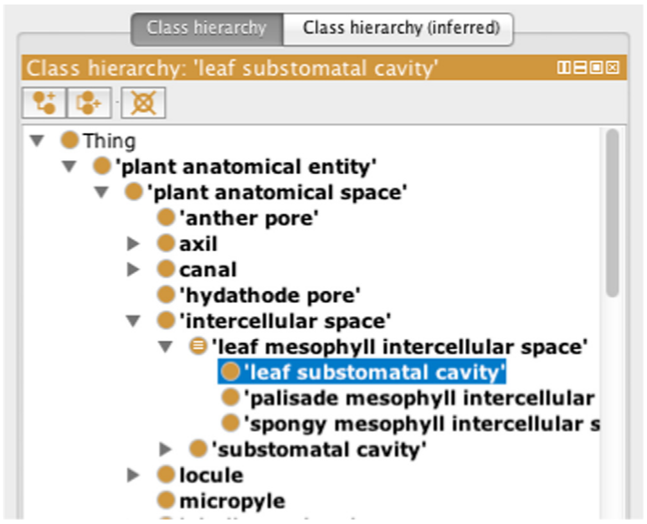

Class hierarchy
Class hierarchy (inferred)
Class hierarchy: 'leaf substomatal cavity'
Thing
'plant anatomical entity'
'plant anatomical space'
'anther pore'
axil
canal
'hydathode pore'
'intercellular space'
'leaf mesophyll intercellular space'
'leaf substomatal cavity'
'palisade mesophyll intercellular
'spongy mesophyll intercellular s
'substomatal cavity'
locule
micropyle

Fig. 1. Visualisation of the class hierarchy in Protégé 4.3.

4.2.3. Ontology building

One of the main problems encountered by ontology authors is a lack of ways to efficiently build and populate ontologies "*If you want an ontology with* 100–1000 *nodes, Protégé is useful for the higher levels and abstract classes.*" (P7). Those who are computer scientists use programming libraries such as the OWL API to efficiently populate ontologies. "*If it is* 2 *classes I do it manually, if it is* 10 *classes* [*I use*] *Protégé, and if it is* 100 *I do it programmatically*." (P7). On the other hand, curators and those who work closely with domain experts often have template based tools that are given to domain experts who enter information into the templates rather than having to model them directly in the ontology. Then these templates are automatically processed and their content is placed into ontologies. "*We had biologists filling out the templates, then we instantiate lots of axioms with these templates.*" (P6).

Many participants reuse existing ontologies when creating their own ontologies. Consequently, only a few ontologies have to be built from scratch. "*We only add new content to the ontology if no other ontology contains that.*" (P6). To reuse and import existing terms, authors try to match or map their terms with those described in other ontologies (which may use a different name to refer to the same concept). Again, some authors turn to specialised information retrieval systems such as BioPortal "*There is so far no available tool, but we do use some ontology mapping tool which is BioPortal.*" (P11). Aligning and merging multiple ontologies is also common for reuse purposes: "*With high level terms, rather than choosing, I keep both terms and then I make an equivalence statement.* […] *With lower level terms, I catch whichever one from whichever ontology that it is suited for and then I refactor it using Protégé and change the ID about the usages of the other class I was going to obsolete to the new class*" (P8).

4.2.4. Reasoning

While frequent automatic classification is often used to prevent the spread of errors, the size of an ontology and the complexity of the axioms cause problems when running the reasoner. "*I don't like to work with ontologies that are so big or complicated that I can't reason reasonably quickly*." (P15). The following strategies are often used to speed up the otherwise lengthy reasoning process:

- Externalising reasoning by employing continuous integration was reported by those working with large ontologies (i.e., curators). "*Reasoning is so memory intensive we have to do all the reasoning separately*." (P12).
- Complex entities are taken out of the ontology and reasoning is done separately. "*The disjoints are kept in a separate file and they are all combined in this set of checks that run every time*" (P12).
- As reported by the participants, using low expressivity languages has some positive consequences: these tend to be more human readable, less error prone and faster to reason. "*The ontology was RDF so it didn't have high expressivity. If you use OWL it is likely that you have unsatisfiable classes if you start using disjointness in conjunction with equivalence, subclass axioms, conjunctions, etc*." (P2).

Those whose preferred environment does not support reasoning turn to Protégé for that. "*I transform my document into OWL and then I will load it with Protégé and classify it*." (P6). Considering that our participants sample worked mainly in the biomedical domain, this strategy was especially reported by those whose principal working environment was OBO-Edit and transformed OBO documents into OWL before running the reasoner.

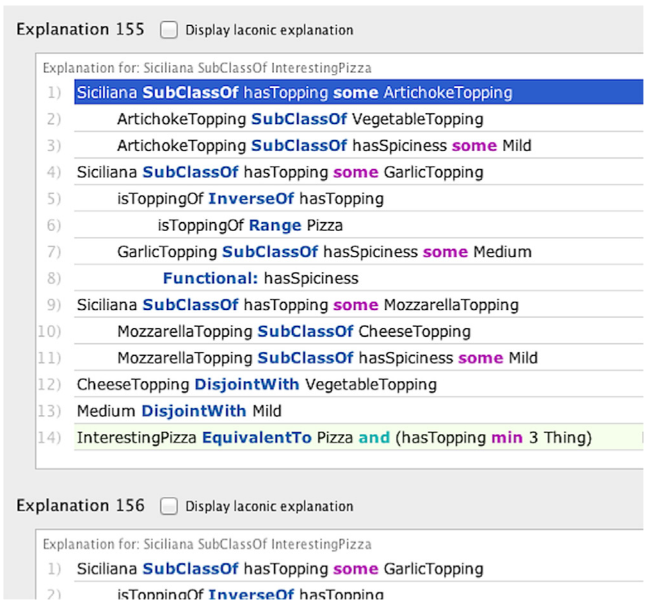
4.2.5. *Debugging and preventing errors*

Despite the presence of an explanation panel in the Protégé editor (see Fig. 2), debugging is still felt to be one of the most challenging tasks, due to a lack of relevant information and appropriate instructions on how to fix errors when they arise. "*If the ontology is big enough, it is very difficult to work out why particular axioms are not working*." (P8).

Debugging is closely connected with the use of a reasoner as certain errors, such as unsatisfiable classes and ontology inconsistency, can only be detected through classification. Frequent invocation of the reasoner is one of the main strategies employed to prevent the spread of errors. Five authors reported classifying at every modification: "*Every axiom change, every class addition, every refactoring, I always run the reasoner and then commit if it is consistent* […] *it is really difficult to find if you make half a dozen changes as it can change the whole topology of the ontology and you cannot understand why*." (P8). Six authors reported that they classify the ontology at a predefined number of additions: "*I classify almost every two axioms because I'm afraid of consequences*." (P5); "*I would add* 5 *terms and then classify*." (P14); "5-10 *axioms and then I'll classify just to check*." (P1). On the other hand, two authors stated that they only classified when complex axioms are added: "*When I add very complex concepts* [*such as*] *existential restriction, universal restriction, transitive, symmetric*. [*If I classify*] *every time I add a concept or a property that would be very foolish because if I'm adding a very simple class or property I'm not adding any disjointness and not adding any functional properties, then the chances of the inconsistency are very low*." (P13).

In addition to frequent classification, the study subjects reported the injection of test axioms or using description logic (DL) queries: "*You build the test classes so you can break down a class, you use a DL query to check which one stops working*." (P6). A set of strategies describe how authors figure out to alleviate the poor feedback given by explanations:

- Fixing first those entities which are more prone to propagate problems. Removing these first is thought to increase the chances of repairing multiple errors at the same time."*Go to the most general inconsistent concept and try to figure out the most common concepts of inconsistencies*." (P9).
- Users try to fix the ontology by making changes in a trial and error fashion, almost randomly. "*It's little bit trial and error, we don't really have a methodology*." (P9).
- Users try to reason about what led them to generate errors by narrowing down the problem. "*If I get no entity under the CheesePizza it means that either I have described pizzas in a wrong way or the CheesePizza is not accurate enough to get that classification*." (P2). Some participants are aware of the authoring mistakes they commit on a regular basis. "*I have some knowledge about my modelling behaviour so I know what sort of things I tend to do wrong, and I check these things very quickly*." (P5).



```
Explanation 155   ☐ Display laconic explanation

Explanation for: Siciliana SubClassOf InterestingPizza
 1) Siciliana SubClassOf hasTopping some ArtichokeTopping
 2)     ArtichokeTopping SubClassOf VegetableTopping
 3)     ArtichokeTopping SubClassOf hasSpiciness some Mild
 4) Siciliana SubClassOf hasTopping some GarlicTopping
 5)     isToppingOf InverseOf hasTopping
 6)         isToppingOf Range Pizza
 7)     GarlicTopping SubClassOf hasSpiciness some Medium
 8)         Functional: hasSpiciness
 9) Siciliana SubClassOf hasTopping some MozzarellaTopping
10)     MozzarellaTopping SubClassOf CheeseTopping
11)     MozzarellaTopping SubClassOf hasSpiciness some Mild
12) CheeseTopping DisjointWith VegetableTopping
13) Medium DisjointWith Mild
14) InterestingPizza EquivalentTo Pizza and (hasTopping min 3 Thing)

Explanation 156   ☐ Display laconic explanation

Explanation for: Siciliana SubClassOf InterestingPizza
 1) Siciliana SubClassOf hasTopping some GarlicTopping
 2)     isToppingOf InverseOf hasTopping
```

Fig. 2. Explanation tool in Protégé 4.3.

4.2.6. *Evaluation*

In the context of ontology authoring, evaluation entails checking whether the ontology represents what is expected, i.e. whether it contains expected entailments and can be used to correctly answer certain questions. It was felt that this quality assurance stage is poorly supported due to the lack of suitable tools and processes. "*We don't have a formal mechanism for checking. We rely on the fact that the ontology is being used*." (P14). Only those who work in larger teams where the ontology authoring process is carried out in a more systematic way employ evaluation procedures. Consulting experts, feedback from users, and using domain specific databases are the main strategies employed for evaluation purposes:

- Experts are consulted when the ontology authors need to be reassured about the decisions made to build the ontology: "*It's reasonable to have experts to confirm what is correct or that is wrong*." (P1).
- In order to palliate the lack of support offered by tools, ontology authors formulate natural language statements about what is expected from an ontology. These statements are often formalised as *competency questions*. "*We have some competence questions for the* [*entity of a domain*] […] *we ask ourselves what*

kind of questions we can ask so that we can develop standardised [*entity of a domain*].” (P11).

5. Discussion

We first consider the tensions with regard to how ontologies are modelled and how these different approaches may be affected by tool support. We then focus on the direct implications the statements given by the interviewees have for the design of ontology authoring tools, and contrast, where appropriate, the ontology authors' requirements and demands with the current state of tool support. We provide a list of operable design recommendations that may help to satisfy the users' needs. Then, we analyse how state-of-the-art tools meet the identified design recommendations. Finally, we discuss the weaknesses of the study and future research work.

5.1. The tensions discovered

The study participants largely consider definition-driven ontologies to have several benefits over manually maintained hierarchies. Maintainability, which has been reported to be one of the main advantages, is a desirable quality, in particular of large ontologies. It was felt by our interviewees that current tools drive authors to manually assert deep hierarchical ontologies, in which users (domain experts or other authors) may find it difficult to navigate, which leads to disorientation. Building definition-driven ontologies is challenging, and authors complain about the lack of tool support. This lack of support and the difficulty of the approach calls for tools that do not only support definition-driven development, but also may ease its adoption for those authors who are less skilled. In any case, tools should cater for different building styles without imposing one over the other.

5.2. Implications for tool design

The results from our interview study indicate that tools such as Protégé 4.* fail to support many of the tasks the interviewees wish to undertake. It appears that existing tools are primarily used for building small ontologies (often as part of a larger ontology) and to visualise medium-sized ontologies that have been built using other means. Assistance for debugging is limited and not sufficiently supported by tools, while evaluation is carried out outside the authoring process. Our findings suggest that the following design insights should enable a better tool support of the ontology authoring process. Note that in many cases tools and techniques for the given purposes already exist; however, based on the statements made in the interviews, we find that existing tools are either unknown to users, do not meet their requirements, or cannot be incorporated seamlessly into existing workflows.

The strategies we describe in Section 4.2 are behavioural markers of cognitive processes that indicate problematic interactions. Consequently, the adoption of the following design recommendations, which derive from the identified strategies, may prevent the problems from happening.

5.2.1. Exploring the ontology

Provide situational awareness. Situational awareness is defined as the extent to which an interactive system allows the user to perceive *relevant* elements of the environment and to comprehend how these elements are integrated into a meaningful whole (Endsley, 1988). Existing ontology authoring tools do not clearly show the consequences of the actions taken, such as the effects of using the *undo* command. When the modification of a single axiom can affect the underlying structure of an entire ontology that contains hundreds of thousands of classes, support for situational awareness is crucial. This may be obtained through textual and visual feedback: the former would inform about the problems that may have arisen (background reasoning would be necessary for this), while the latter could be achieved, for example, by highlighting the updated elements.

Overview, filtering, and linking features. Navigating ontologies is prone to information overload and disorientation. This is especially true of the larger and axiomatically richer ontologies. While there exist tools such as OWLViz in Protégé, the study participants did not consider it to be suitable for the task of navigating and understanding ontologies: “[*OWLViz*] *only shows the class graph which is almost the same as the hierarchy*” (P4); “*What I use for* [*browsing*] *big ontologies is the class hierarchy*, *the usage and then … for more complex tasks is use OPPL*” (P2). Ontology visualisation and summary is an active area of research, but as yet no tools have had a wide acceptance and they are not easily integrated or interoperable with the main editing tools. Implementing fundamental principles of information visualisation by providing improved overviews and filters could alleviate this problem. Overviews may help those who are not familiar with the domain, whereas filters will speed up the navigation of domain experts. Considering the interlinked nature of the elements of an ontology, hyperlinking the class hierarchy, entities, and axioms may help authors to use the information scent in links to traverse the ontology.

5.2.2. Ontology building

Efficient ontology population methods. Current tools fail to support the often large numbers of additions and modifications that need to be made to large ontologies, and ontology developers generally fall back onto other methods, such as programmatic population of ontologies. This shortfall calls for the capabilities in an ontology development tool to efficiently populate ontologies from existing sources. Furthermore, spreadsheets and templates for populating ontologies that abstract from the OWL representation are especially relevant if domain experts are involved in the building process.

Support for ontology retrieval and reuse. Searching for existing ontologies and terms in order to import them requires information retrieval systems that do not only *retrieve* those resources that match a query, but also allow authors to select and incorporate (specific parts of) the results into the working ontology. Ontology repositories such as the NCBO BioPortal (Noy et al., 2009) already allow users to browse ontologies and query for specific terms across the repository in order to find suitable ontologies for reuse. In order to deal with discrepancies between the working ontology and the reused ones which may arise, for example, from the use of different terms, tools should provide mechanisms for mapping and aligning ontologies. Furthermore, modularisation may be used to extract suitable ontology subsets which can be merged into the working ontology while avoiding errors and keeping up-to-date with changes in the imported ontology and its fragments; however, choosing a suitable signature for extracting modules is known to be a challenging task for ontology developers (Del Vescovo et al., 2010).

5.2.3. Reasoning, debugging and evaluation

On-the-fly reasoning. The (perceived) complexity of OWL leads some authors to invoke the reasoner at every modification they make, or when complex modifications are made that are believed to potentially cause an error. Some authors adopt this frequent classification strategy in order to avoid the propagation of problems. Considering the high demands of reasoning, tools and reasoners should allow authors to reason over subsets of ontologies and axioms to speed up the process. Ideally, authoring tools should also provide background reasoning capabilities, as do the compilers of software development IDEs. *Incremental* reasoning, which is already supported by some OWL reasoners (Cuenca Grau et al., 2010; Kazakov and Klinov, 2013), seems like a promising approach to providing such background reasoning capabilities.

Debugging. Debugging has emerged as one of the most cumbersome tasks for ontology authors, who, despite the presence of explanation facilities in Protégé, frequently struggle with understanding what steps to take in order to repair an ontology. This problem is often dealt with by first identifying those explanations the authors consider to be the most relevant (e.g. those using the most high-level terms), as they expect them to have the biggest repair impact. However, it is clear that ontology developers require more sophisticated debugging support which alleviates the problem of dealing with multiple explanations, and provides guidance to help users choose the best possible repair strategy. While there has been a significant body of research in this direction (e.g. Kalyanpur et al., 2005; Horridge et al., 2011; Bail et al., 2011; Nguyen et al., 2012), most of these improved debugging features have yet to make it into current mainstream OWL editors.

Evaluation and testing We found that, while testing is poorly integrated in the ontology development process, evaluation is done outside this process. Unit tests in the shape of competency questions, as suggested by Vrandecic and Gangemi (2006), would allow ontology authors to compare their artefacts against initial domain requirements in terms of coverage and capabilities. However, one problem that arises from competency questions is that the answers need to be *known* in order to evaluate the correctness of the ontology; it is clear that it is hardly possible to create complete competency questions that test all the information in an ontology. Ren et al. (2014) have found that patterns exist between different domains and have shown it is feasible to build templates for competency questions using these patterns.

Established methodologies indicate that involving users, domain experts and ontology engineers provides a better coverage of competence questions (Suárez-Figueroa and Gómez-Pérez, 2012). However, getting together all the actors may not always be possible, especially if an agile development paradigm is followed. In order to deal with *non-entailments*, i.e. desirable statements that the users expect to be entailed by an ontology but are not entailed, ontology completion techniques (e.g. (Baader et al., 2007)) could be integrated into an ontology development environment to make (both positive and negative) non-entailments visible.

5.3. From design insights to design recommendations

In Section 5.2 we enumerate a number of design guidelines that emerge from the strategies we discuss in Section 4.2. We summarise these and shape them as more specific operable design recommendations that may prevent the described problems from occurring.

- Increase situational awareness by giving feedback about the consequences of actions carried out has been reported to be crucial, especially when axioms are modified, commands such as *undoing* are invoked and when the reasoner is run. Providing a history of modifications to which users can revert may alleviate this problem.
- For those who are not familiar with the ontology or for large ontologies, provide overviews of the selected entity of the hierarchy and summarise its axiomatic complexity in order to facilitate its exploration. For those who are familiar with the ontology provide bookmarks, landmarks and shortcuts to easily navigate to a specific entity.
- Ease the navigation to those who are looking for a particular entity through linking mechanisms that associate different entities, and provide search and filtering features for information seeking.
- Include support for not only searching across remote ontologies, but also to retrieve entities and incorporate them into the working ontology. Providing mechanisms for mapping and merging ontologies in this process is of utmost importance.
- Incorporate design templates and spreadsheets to efficiently populate ontologies and mitigate the daunting task of adding a high number of entities.
- Provide on-the-fly reasoning capabilities so that incremental reasoning is done as a background process.
- Remove information overload of explanations by identifying key explanations and provide filters to better grasp explanations in order to support debugging.
- Include predefined unit tests and templates to generate competence questions in order to support the evaluation process.

5.4. Support by state-of-the-art tools

In order to draw a clear picture of the functionalities and shortfalls of current OWL tool support, we selected a sample of OWL ontology authoring tools listed in http://semanticweb.org/wiki/Tools and assessed them against the design recommendations identified in Section 5.3: Protégé 4.3,[7] Web Protégé (Tudorache et al., 2013), the free edition of TopBraid Composer, and SWOOP 2.3 (Kalyanpur et al., 2006), which is no longer actively maintained. Our goal is not to run a comprehensive analysis of all existing tools, but to suggest the usefulness of the identified design recommendations as a framework to evaluate ontology authoring tools. In any case, our sample accounts for the tools used by more than 83% of ontology authors (Cardoso, 2007), which allows to assess the situation of the current landscape of tools.

As expected, the analysis in Table 3 suggests that tool support is limited, which is in line with our findings. That is, none of the tools of our sample fully meet all (or even half) the design recommendations. The fact that each recommendation is (partially) met by at least one tool indicates that it is feasible to have all the desired functionalities. Tools stand out for particular features: Protégé is appropriate for having an overview of ontology, and merging and aligning ontologies; WebProtégé is suitable for searching and retrieving ontologies and their entities into the current workspace; TopBraid has a potentially good support for testing and evaluating ontologies; finally SWOOP has efficient error repair functionalities in addition to providing users with situational awareness. Therefore, tools have mostly weaknesses and a few strengths, which makes them difficult to use at some stages of the authoring process.

The non-existence of tools that provide users with support for all the stages of the ontology authoring lifecycle suggests that we are in the same situation described by Katifori et al. (2007) and Dzbor et al. (2006) a few years ago. While tools have incorporated new features, they still fail to keep up with the evolved needs of users, which are addressed by a repertoire of sophisticated strategies.

5.5. Caveats and future work

Due to their subjectivity, the outcomes obtained through direct enquiry methods such as the interviews we discuss in this paper are of an exploratory nature. As such, they may not be generalisable, although they are an invaluable picture of the problems encountered by what we consider to be typical, experienced

[7] In this analysis we consider the default downloadable version of Protégé 4.3 without external plugins.

Table 3
Support by current tools: ✓ indicates full support, ∼ partial support, and lack of support is denoted by ×.

	Protégé	**WebProtégé**	**TopBraid**	**SWOOP**
Exploring ontologies				
Situational awareness	× Poorly supported: the mechanisms provided include buttons that allow to navigate back and forth in the interaction history and the 'undo' functionality	∼ While one cannot visualise the consequences of actions, every single entity has a history describing all the modifications. However, there is no way to get to a previous state of the ontology. These modifications get to the authors not only through the web application, but also through RSS or email	× History navigation functionality (back & forth) and 'undo' provided	✓ Buttons that allow to navigate back and forth in the interaction history. The 'undo' functionality works together with a change tracker that allows the user to restore the ontology to a previous state. The consequences of reverting to a previous state are given in detail
Overview	✓ The description of each entity is shown on a separate window; ontology metrics are shown	∼ Only ontology metrics are shown	∼ The ontology overview contains the namespaces and some statistics of the current ontology	∼ The ontology info tab shows some metrics and statistics about the ontology
Search, filtering and linking	✓ Search functionality and linking provided	∼ No search features and considering it is a web application, a heavier linking would have been expected. For instance, individuals, IRIs, namespaces, ranges, and domains cannot be clicked	∼ Search functionality provided. It searches across the ontologies that are currently loaded	✓ Search functionality and linking provided
Ontology building				
Ontology retrieval and reuse	×	✓ Not only it allows to search and retrieve terms from other ontologies, but also from ontology repositories such as BioPortal	×	×
Efficient population	∼ Although not included in the default version, there are several plug-ins that allow the import of datasets from spreadsheets, CSV files, XML, or relational databases	×	×	×
Support for merging and aligning	✓ Support for merging and refactoring	×	∼ Refactoring functionalities provided	×
Debugging and evaluation				
On-the-fly reasoning	×	× There are no online reasoning capabilities. Reasoning should be done locally	×	∼ Support for extracting modules and reason separately
Efficient error repair	× Explanations are not prioritised	×	×	✓ Explanations are classified by arity, impact, usage, and rank (the metric that computes rank can be customised)
Evaluation and testing	×	×	✓ Predefined SPARQL templates are provided	×

ontology authoring tool users. When selecting participants we targeted ontologists; however, OWL and Protégé came out as the most significant technologies. Moreover, the ontology authors who reported the use of other languages and tools such as OBO and OBO-Edit stressed the strengths and weaknesses of OWL and Protégé. Also, the fact that a majority of participants belong to the life sciences polarises our results towards the particular characteristics (and challenges) of life science ontologies.

Finally, all participants in the study had several years of experience in working with ontologies, which means that the study may not be indicative of the problems faced by ontology *novices*. It is clear that users who are new to ontologies as a formalism might struggle with additional problems to the ones we identified here such as understanding OWL's semantics, Rector et al., 2004; on the other hand, however, we argue that improved tool support for existing ontologies users will also entail benefits for novices. For the near future, we will run laboratory tests using monitoring tools in order to explore the extent and scope of the strategies reported by participants; a preliminary analysis can be seen in Vigo et al. (2014b). We also plan to extend the study to different categories of ontology developers and users.

6. Conclusion

In this paper, we explored the problems encountered by ontology authors in the ontology engineering process, as well as the strategies they employ in order to deal with those problems. We presented the results of an interview study with 15 participants and identified design recommendations for ontology development based on the insights gained from the study.

Our interviews suggest that ontology authors have moved beyond Protégé's capabilities, the tool used by the vast majority of participants. The ontology authors interviewed for this study build ontologies in a way that makes more demands on the tool than the tool can deliver. Sophistication of the tools has not kept pace with the sophistication of the users. Indeed, Protégé 4.⁎ (the version of Protégé used by these participants) is now quite an old tool—the Protégé OWL extension was first released around the same time OWL became a W3C recommendation in 2004. It is time for a step change in ontology development environments to match both the users' sophistication and the advances in ontology engineering, while catering for a wide variety of users and building styles.

As ontologies are increasingly used to model and integrate information from various domains of knowledge, adequate tool support is crucial to ensure the quality and permanence of ontology engineering efforts. Moreover, accessible tools may also lower the barriers for novice users, thus supporting the acceptance and uptake of Semantic Web technologies.

Acknowledgments

This research has been funded by the EPSRC project: *WhatIf: Answering "What if…" questions for Ontology Authoring*. EPSRC reference EP/J014176/1.

References

Baader, F., Ganter, B., Sertkaya, B., Sattler, U., 2007. Completing description logic knowledge bases using formal concept analysis. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07), pp. 230–235.

Bail, S., Horridge, M., Parsia, B., Sattler, U., 2011. The justificatory structure of the NCBO bioportal ontologies. In: Proceedings of the 10th International Semantic Web Conference (ISWC-11).

Braun, V., Clarke, V., 2006. Using thematic analysis in psychology. Qual. Res. Psychol. 3 (2), 77–101.

Buhrmester, M., Kwang, T., Gosling, S.D., 2011. Amazon's mechanical turk: a new source of inexpensive, yet high-quality, data?. Perspect. Psychol. Sci. 6 (1), 3–5.

Cardoso, J., 2007. The semantic web vision: where are we?. IEEE Intell. Syst. 22 (5), 84–88.

Cimino, J.J., 1998. Desiderata for controlled medical vocabularies in the twenty-first century. Methods Inf. Med. 37 (4-5), 394.

Cuenca Grau, B., Halaschek-Wiener, C., Kazakov, Y., Suntisrivaraporn, B., 2010. Incremental classification of description logics ontologies. J. Autom. Reason. 44 (4), 337–369.

Del Vescovo, C., Parsia, B., Sattler, U., Schneider, T., 2010. The modular structure of an ontology: an empirical study. In: Proceedings of the 23rd International Workshop on Description Logics (DL-10).

Duineveld, A., Stoter, R., Weiden, M., Kenepa, B., Benjamins, V., 2000. WonderTools? A comparative study of ontological engineering tools. Int. J. Hum. Comput. Stud. 52 (6), 1111–1133, URL ⟨http://www.sciencedirect.com/science/article/pii/S107158199990366X⟩.

Dzbor, M., Motta, E., Aranda, C.B., Perez, J.M.G., Goerlitz, O., Lewen, H., 2006. Developing ontologies in OWL: an observational study. In: OWL: Experiences and Directions Workshop.

Endsley, M., 1988. Situation awareness global assessment technique (SAGAT). In: Proceedings of the IEEE 1988 National Aerospace and Electronics Conference, 1988, NAECON 1988, vol. 3, pp. 789–795.

Hepp, M., 2007. Possible ontologies: how reality constrains the development of relevant ontologies. IEEE Internet Comput. 11 (1), 90–96.

Horridge, M., Bail, S., Parsia, B., Sattler, U., 2011. The cognitive complexity of OWL justifications. In: Proceedings of the 10th International Semantic Web Conference (ISWC-11), Lecture Notes in Computer Science, vol. 7031, pp. 241–256. URL ⟨http://www.dx.doi.org/10.1007/978-3-642-25073-6_16⟩.

Kalyanpur, A., Parsia, B., Sirin, E., Cuenca Grau, B., Hendler, J., 2006. Swoop: a web ontology editing browser. J. Web Semant. 4 (2), 144–153.

Kalyanpur, A., Parsia, B., Sirin, E., Hendler, J., 2005. Debugging unsatisfiable classes in OWL ontologies. J. Web Semant. 3 (4), 268–293.

Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., Giannopoulou, E., 2007. Ontology visualization methods—a survey. ACM Comput. Surv. 39 (4), URL ⟨http://doi.acm.org/10.1145/1287620.1287621⟩.

Kazakov, Y., Klinov, P., 2013. Incremental reasoning in OWL EL without bookkeeping. In: Proceedings of the 12th International Semantic Web Conference (ISWC-13).

Khalili, A., Auer, S., 2013. User interfaces for semantic authoring of textual content: a systematic literature review. Web Semantics: Science, Services and Agents on the World Wide Web. URL ⟨http://www.sciencedirect.com/science/article/pii/S1570826813000498⟩.

Lam, J.S.C., 2007. Methods for Resolving Unsatisfiable Ontologies (Ph.D. thesis), University of Aberdeen.

Lambrix, P., Habbouche, M., Pérez, M., 2003. Evaluation of ontology development tools for bioinformatics. Bioinformatics 19 (12), 1564–1571, URL ⟨http://bioinformatics.oxfordjournals.org/content/19/12/1564.abstract⟩.

Landis, J.R., Koch, G.G., 1977. The measurement of observer agreement for categorical data. Biometrics 33 (1), 159–174, URL ⟨http://www.jstor.org/stable/2529310⟩.

McGuinness, D.L., Patel-Schneider, P.F., 1998. Usability issues in knowledge representation systems. In: Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98), pp. 608–614.

Nguyen, T.A.T., Power, R., Piwek, P., Williams, S., 2012. Measuring the understandability of deduction rules for OWL. In: Proceedings of WoDOOM-12.

Noy, N.F., Shah, N.H., Whetzel, P.L., Dai, B., Dorf, M., Griffith, N., Jonquet, C., Rubin, D. L., Storey, M.-A., Chute, C.G., Musen, M.A., 2009. Bioportal: ontologies and integrated data resources at the click of a mouse. Nucleic Acids Res. 37, W170–W173.

Randall, D., Procter, R., Lin, Y., Poschen, M., Sharrock, W., Stevens, R., 2011. Distributed ontology building as practical work. Int. J. Hum. Comput. Stud. 69 (4), 220–233, URL ⟨http://www.sciencedirect.com/science/article/pii/S1071581911000024⟩.

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., Wroe, C., 2004. Owl pizzas: practical experience of teaching OWL-DL: common errors & common patterns. In: Motta, E., Shadbolt, N., Stutt, A., Gibbins, N. (Eds.), Engineering Knowledge in the Age of the Semantic Web. Vol. 3257 of Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 63–81. URL ⟨http://dx.doi.org/10.1007/978-3-540-30202-5_5⟩.

Rector, A.L., 2003. Modularisation of domain ontologies implemented in description logics and related formalisms including OWL. In: Proceedings of the 2nd International Conference on Knowledge Capture. K-CAP '03. ACM, New York, NY, USA, pp. 121–128. URL ⟨http://doi.acm.org/10.1145/945645.945664⟩.

Ren, Y., Parvizi, A., Mellish, C., Pan, J., van Deemter, K., Stevens, R., 2014. Towards competency question-driven ontology authoring. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (Eds.), The Semantic Web: Trends and Challenges. Vol. 8465 of Lecture Notes in Computer Science. Springer International Publishing, pp. 752–767. URL ⟨http://www.dx.doi.org/10.1007/978-3-319-07443-6_50⟩.

Suárez-Figueroa, M., Gómez-Pérez, A., 2012. Ontology requirements specification. In: Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (Eds.), Ontology Engineering in a Networked World. Springer, Berlin Heidelberg, pp. 93–106, URL ⟨http://dx.doi.org/10.1007/978-3-642-24794-1_5⟩.

Suárez-Figueroa, M.C., Gómez-Pérez, A., Fernández-López, M., 2012. The NeOn methodology for ontology engineering. In: Suárez-Figueroa, M.C., Gómez-Pérez, A., Motta, E., Gangemi, A. (Eds.), Ontology Engineering in a Networked World. Springer, Berlin Heidelberg, pp. 9–34, URL ⟨http://dx.doi.org/10.1007/978-3-642-24794-1_2⟩.

Tudorache, T., Nyulas, C., Noy, N.F., Musen, M.A., 2013. WebProtégé: a collaborative ontology editor and knowledge acquisition tool for the web. Semant Web 4 (1), 89–99.

Vigo, M., Jay, C., Stevens, R., 2014a. Design insights for the next wave ontology authoring tools. In: Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems. CHI '14. ACM, New York, NY, USA, pp. 1555– 1558. URL ⟨http://doi.acm.org/10.1145/2556288.2557284⟩.

Vigo, M., Jay, C., Stevens, R., 2014b. Protégé4US: Harvesting ontology authoring data with Protégé. In: Workshop on Human–Semantic Web Interaction, HSWI '14. URL ⟨https://www.escholar.manchester.ac.uk/uk-ac-man-scw:223543⟩.

Vrandecic, D., Gangemi, A., 2006. Unit tests for ontologies. In: Proceedings of the OTM 2006 Workshops, vol. 2, pp. 1012–1020.

Wang, H., Tudorache, T., Dou, D., Noy, N.F., Musen, M.A., 2013. Analysis of user editing patterns in ontology development projects. In: On the Move to Meaningful Internet Systems: OTM 2013 Conferences, Lecture Notes in Computer Science, vol. 8185, pp. 470–487. URL ⟨http://www.dx.doi.org/10.1007/978-3-642-41030-7_34⟩.

Warren, P., 2013. Ontology Users' Survey—Summary of Results. Technical Report KMI-13-1, Knowledge Media Institute.

Warren, P., Mulholland, P., Collins, T., Motta, E., 2014. The usability of description logics. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (Eds.), The Semantic Web: Trends and Challenges, Lecture Notes in Computer Science, vol. 8465. Springer International Publishing, pp. 550-564. URL ⟨http://www.dx.doi.org/10.1007/978-3-319-07443-6_37⟩.