A Novel Approach to Visualizing and Navigating Ontologies

Enrico Motta[1], Paul Mulholland[1], Silvio Peroni[2], Mathieu d'Aquin[1], Jose Manuel Gomez-Perez[3], Victor Mendez[3], Fouad Zablith[1]

[1]Knowledge Media Institute, The Open University, MK7 6AA, Milton Keynes, UK
{e.motta, p.mulholland, m.daquin, f.zablith}@open.ac.uk

[2]Dept. of Computer Science, University of Bologna, 40127 Bologna, Italy
speroni@cs.unibo.it

[3]Intelligent Software Components (iSOCO) S.A., 28042 Madrid Spain
{jmgomez, vmendez}@isoco.com

Abstract. Observational studies in the literature have highlighted low levels of user satisfaction in relation to the support for ontology visualization and exploration provided by current ontology engineering tools. These issues are particularly problematic for non-expert users, who rely on effective tool support to abstract from representational details and to be able to make sense of the contents and the structure of ontologies. To address these issues, we have developed a novel solution for visualizing and navigating ontologies, *KC-Viz,* which exploits an empirically-validated ontology summarization method, both to provide concise views of large ontologies, and also to support a 'middle-out' ontology navigation approach, starting from the most information-rich nodes (*key concepts*). In this paper we present the main features of KC-Viz and also discuss the encouraging results derived from a preliminary empirical evaluation, which suggest that the use of KC-Viz provides performance advantages to users tackling realistic browsing and visualization tasks. Supplementary data gathered through questionnaires also convey additional interesting findings, including evidence that prior experience in ontology engineering affects not just objective performance in ontology engineering tasks but also subjective views on the usability of ontology engineering tools.

Keywords: Ontology Visualization, Key Concepts, Ontology Summarization, Ontology Navigation, Ontology Engineering Tools, Empirical Evaluation.

1 Introduction

Browsing ontologies to make sense of their contents and organization is an essential activity in ontology engineering. This is particularly the case today, as the significant increase in the number of ontologies available online means that ontology engineering projects often include a reuse activity, where people first locate ontologies which may be relevant to their project – e.g., by using ontology search engines, such as Sindice [1] or Watson [2], and then examine them to understand to what extent they provide solutions to their modelling needs.

In addition, ontologies are no longer developed and used exclusively by specialized researchers and practitioners. On the contrary, as ontologies are increasingly used in a variety of scenarios, such as research, healthcare, and business, more and more domain experts and other relatively inexperienced users are involved in the ontology engineering process, especially in the context of community-wide ontology development activities [3].

However, evidence gathered through observational studies [4] indicates low levels of user satisfaction with the tool support currently available to users for visualizing and navigating ontologies, in particular in relation to the lack of effective mechanisms for ‘content-level visualization’, including support for selective visualization of ontology parts, summaries, and overviews [4]. Needless to say, these problems affect in particular inexperienced users, who rely on effective tool support to abstract from representational details and make sense of the contents and the structure of ontologies.

Attempting to address these issues, we have developed a novel solution for visualizing and navigating ontologies, *KC-Viz,* which builds on our earlier work on *key concepts extraction* [5], both as a way to provide concise overviews of large ontologies, and also to support a ‘middle-out’ ontology navigation approach, starting from the most information-rich nodes[1] (*key concepts*). Building on its ability to abstract out from large ontologies through key concept extraction, KC-Viz provides a rich set of navigation and visualization mechanisms, including flexible *zooming* into and *hiding* of specific parts of an ontology, *history* browsing, saving and loading of customized *ontology views*, as well as essential interface customization support, such as graphical zooming, font manipulation, tree layout customization, and other functionalities. KC-Viz is a core plugin of the NeOn Toolkit and can be downloaded from http://neon-toolkit.org.

In this paper we introduce KC-Viz and we present the results from a preliminary empirical evaluation, which suggest that the use of KC-Viz provides performance advantages to users tackling realistic browsing and visualization tasks. Moreover, we also report on additional findings gathered through questionnaires, which offer a number of other insights, including evidence that prior experience in ontology engineering affects not just objective performance in ontology engineering tasks but also subjective views on the usability of ontology engineering tools.

2 Approaches to visualizing and navigating ontologies

2.1 Literature Review

The issue of how best to support visualization and navigation of ontologies has attracted much attention in the research community. As Wang and Parsia emphasize [6], “effective presentation of the hierarchies can be a big win for the users”, in particular, but not exclusively, during the early stages of a *sensemaking*[2] process, when a user is trying to build an initial mental model of an ontology, focusing less on

[1] In the paper we will use the terms ‘node’, ‘concept’, and ‘class’ interchangeably to refer to classes in an ontology.

[2] In the rest of the paper we will use the term ‘sensemaking’ to refer to a specific ontology engineering task, where the user is primarily concerned with understanding the contents and overall structure of the ontology, i.e., acquiring an overview of the concepts covered by the ontology and the way they are organized in a taxonomy.

specific representational details than on understanding the overall organization of the ontology. In particular, as discussed in [7], there are a number of functionalities that an effective visualization system needs to support, including (but not limited to) the ability to provide *high level overviews* of the data, *to zoom in* effectively on specific parts of the data, and *to filter out* irrelevant details and/or irrelevant parts of the data.

An approach to addressing the issue of providing high level overviews of hierarchical structures focuses on maximizing the amount of information on display, through *space-filling* solutions, such as those provided by *treemaps* [8]. Treemaps have proved to be a very successful and influential visualization method, used not just to represent conceptual hierarchies but also to visualize information in several mainstream sectors, including news, politics, stock market, sport, etc. However, while treemaps define a clever way to provide concise overviews of very large hierarchical spaces, they are primarily effective when the focus is on leaf nodes and on a particular dimension of visualization, in particular if colour-coding can be used to express different values for the dimension in question. However, as pointed out in [6], treemaps are not necessarily effective in supporting an understanding of topological structures, which is what is primarily needed in the ontology sensemaking context highlighted earlier.

State of the art ontology engineering toolkits, such as Protégé[3] and TopBraid Composer[4], include visualization systems which use the familiar *node-link diagram* paradigm to represent entities in an ontology and their taxonomic or domain relationships. In particular, both the OwlViz visualizer in Protégé and the 'Graph View' in TopBraid make it possible for users to navigate the ontology hierarchy by selecting, expanding and hiding nodes. However OwlViz arguably provides more flexibility, allowing the user to customize the expansion radius and supporting different modalities of use, including the option of automatically visualizing in OwlViz the current selection shown in the Protégé Class Browser.

SpaceTree [9], which also follows the *node-link diagram* paradigm, is able to maximize the number of nodes on display, by assessing how much empty space is available. At the same time it also avoids clutter by utilizing informative preview icons. These include miniatures of a branch, which are able to give the user an idea of the size and shape of an un-expanded subtree at a very high level of abstraction, while minimizing the use of real estate.

Like treemaps, CropCircles [6] also uses geometric containment as an alternative to classic node-link displays. However, it tries to address the key weakness of treemaps, by sacrificing space in order to make it easier for users to understand the topological relations in an ontology, including both parent-child and sibling relations. An empirical evaluation comparing the performance of users on topological tasks using treemaps, CropCircles and SpaceTree showed that, at least for some tasks, users of CropCircles performed significantly better than those using treemaps [6]. However, SpaceTree appears to perform significantly better than either treemaps or CropCircles on node finding tasks.

A number of 'hybrid' solutions also exist, such as Jambalaya [10] and Knoocks [11], which attempt to combine the different strengths of containment-based and

---

[3] http://protege.stanford.edu/.

[4] http://www.topquadrant.com/products/TB_Composer.html.

node-link approaches in an integrated framework, by providing both alternative visualizations as well as hybrid, integrated views of the two paradigms.

The group of techniques categorized in [12] as *"context + focus and distortion"* are based on "the notion of distorting the view of the presented graph in order to combine context and focus. The node on focus is usually the central one and the rest of the nodes are presented around it, reduced in size until they reach a point that they are no longer visible" [12]. These techniques are normally based on hyperbolic views of the data and offer a good trade-off – a part of the ontology is shown in detailed view, while the rest is depicted around. A good exemplar of this class of approaches is *HyperTree* [13].

Finally, we should also consider in this short survey the most ubiquitous and least visual class of tools, exemplified by plugins such as the Class Browser in Protégé and the Ontology Navigator in the NeOn Toolkit. These follow the classic file system navigation metaphor, where clicking on a folder opens up its sub-folders. This approach is ubiquitous in both file system interfaces and ontology engineering tools and, in the case of ontologies, it allows the user to navigate the ontology hierarchy simply by clicking on the identifier of a class, to display its subclasses, and so on. While superficially a rather basic solution, especially when compared to some of the sophisticated visual metaphors that can be found in the literature, this approach can be surprisingly effective for two reasons: i) it is very familiar to users and ii) it makes it possible to display quite a lot of information in a rather small amount of space, in contrast with node-link displays, which can be space-hungry. As a result it is not surprising that these interfaces often perform better in evaluation scenarios than the graphical alternatives. For instance, the evaluation reported in [14] shows that subjects using the Protégé Class Browser fared better than those using alternative visualization plugins in a number of ontology engineering tasks.

2.2 Discussion

It is clear from the review in the previous section that different approaches exhibit different strengths and weaknesses and that in general the effectiveness of a particular solution depends on the specific task it is being used for. For example, the evaluation presented in [6] suggests that CropCircles may perform well in 'abstract' topological tasks, such as "Find the class with the most direct subclasses", but SpaceTree appears to be better in locating a specific class. As already mentioned, here we are primarily concerned with the ontology sensemaking task, so what we are looking for is effective support for the user in quickly understanding what are the main areas covered by the ontology, how is the main hierarchy structured, etc.

The problem is a particularly tricky one because, once an ontology is large enough, it is not possible to show its entire structure in the limited space provided by a computer screen and therefore a difficult trade-off needs to be addressed. On the one hand the information on display needs to be coarse-grained enough to provide an overview of the ontology, thus ensuring the user can maintain an overall mental model of the ontology. On the other hand, an exploration process needs to be supported, where the user can effectively home in on parts of the ontology, thus changing the level of analysis, while at the same time not losing track of the overall organization of the ontology. In sum, we can say that the main (although obviously not the only) issue is one of reconciling abstraction with focus.

However, a problem affecting all the approaches discussed in the review is that all of them essentially use geometric techniques to providing abstraction, whether it is the use of a hyperbolic graph, geometric containment, or the miniature subtrees provided by SpaceTree.

In contrast with these approaches, human experts are able to provide effective overviews of an ontology, simply by highlighting the key areas covered by the ontology and the classes that best describe these areas. In particular, the work reported in [5] provides empirical evidence that there is a significant degree of agreement among experts in identifying the main concepts in an ontology, and it also shows that our algorithm for key concept extraction (*KCE*) is also able to do so, while maintaining the same level of agreement with the experts, as they have among themselves [5]. Hence, the main hypothesis underlying our work on KC-Viz is that effective abstraction mechanisms for ontology visualization and navigation can be developed by building on the KCE algorithm, thus going beyond purely geometric approaches and focusing instead on displaying the concepts which are identified as the most useful for making sense of an ontology.

3 Overview of KC-Viz

3.1 Key Concept Extraction

Our algorithm for key concept extraction [5] considers a number of criteria, drawn from psychology, linguistics, and formal knowledge representation, to compute an 'importance score' for each class in an ontology. In particular, we use the notion of *natural category* [15], which is drawn from cognitive psychology, to identify concepts that are information-rich in a psycho-linguistic sense. Two other criteria are drawn from the topology of an ontology: the notion of *density* highlights concepts which are information-rich in a formal knowledge representation sense, i.e., they have been richly characterized with properties and taxonomic relationships, while the notion of *coverage* states that the set of key concepts identified by our algorithm should maximize the coverage of the ontology with respect to its is-a hierarchy[5]. Finally, the notion of *popularity*, drawn from lexical statistics, is introduced as a criterion to identify concepts that are likely to be most familiar to users.

The *density* and *popularity* criteria are both decomposed in two sub-criteria, *global* and *local density*, and *global* and *local popularity* respectively. While the global measures are normalized with respect to all the concepts in the ontology, the local ones consider the relative density or popularity of a concept with respect to its surrounding concepts in the is-a hierarchy. The aim here is to ensure that 'locally significant' concepts get a high score, even though they may not rank too highly with respect to global measures.

Each of the seven aforementioned criteria produces a score for each concept in the ontology and the final score assigned to a concept is a weighted sum of the scores resulting from individual criteria. As described in [5], which provides a detailed account of our approach to key concept extraction, the KCE algorithm has been

[5] By 'is-a hierarchy' here, we refer to the hierarchy defined by rdfs:subClassOf relations.

shown to produce ontology summaries that correlate significantly with those produced by human experts.

3.2 Exploring ontologies with KC-Viz

Normally, a KC-Viz session begins by generating an initial summary of an ontology, to get an initial 'gestalt' impression of the ontology. This can be achieved in a number of different ways, most obviously by i) selecting the ontology in question in the 'Ontology Navigator' tab of the NeOn Toolkit, ii) opening up a menu of options by right clicking on the selected ontology, and then iii) choosing Visualize Ontology → Visualize Key Concepts, through a sequence of menus. Figure 1[6] shows the result obtained after performing this operation on the SUMO ontology, a large upper level ontology, which comprises about 4500 classes. The version used in these examples can be downloaded from http://www.ontologyportal.org/SUMO.owl.

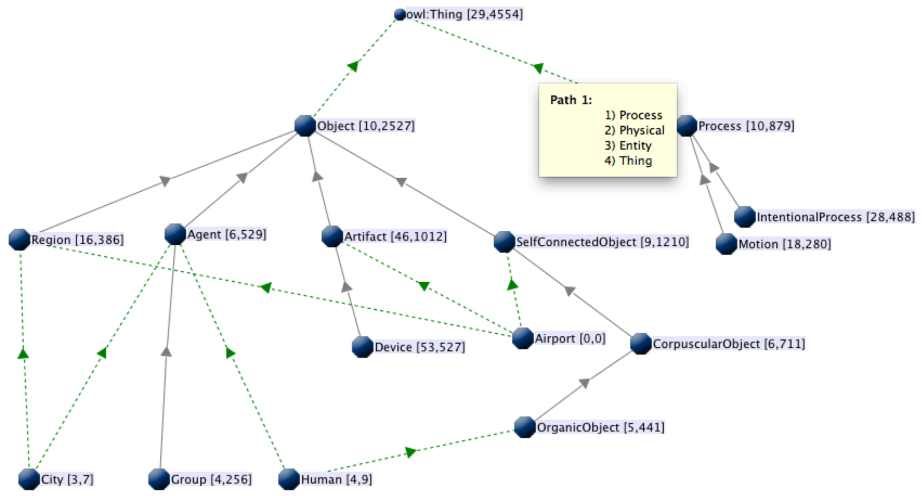


Figure 1. Initial visualization of the SUMO ontology.

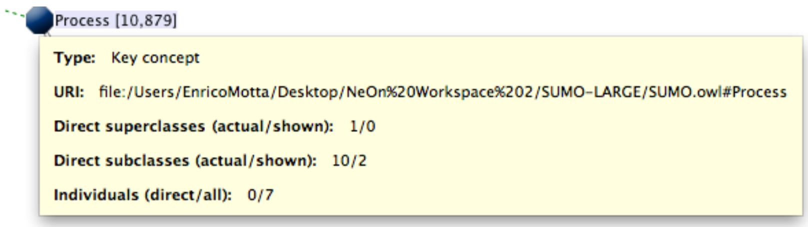
The summary shown in Figure 1, which has been generated by the KCE algorithm, includes 16 concepts because we have set the size of our ontology summary to 15 and the algorithm has automatically added the most generic concept, owl:Thing, to ensure that the visualization displays a connected graph. If we wish to display more or less succinct graphs, we can do so by changing the size of the ontology summary. The solid grey arrows in the figure indicate *direct* rdfs:subClassOf links, while the dotted green arrows indicate *indirect* rdfs:subClassOf links. As shown in the figure, by hovering the mouse over an indirect rdfs:subClassOf links, we can see the chain of rdfs:subClassOf relations, which have been summarized by the indirect link. In this

[6] As shown in Figure 1, KC-Viz is based on the *node-link diagram* paradigm. However, as correctly pointed out by an anonymous reviewer, the KCE algorithm can in principle be used with alternative visualization styles, and indeed this is something we plan to explore in the future. The rationale for adopting the *node-link diagram* paradigm in the first instance is that this is a familiar representation for users and we wish to test our hypothesis that the use of key concepts can succeed in equipping this approach with effective abstraction mechanisms.

case, we can see that an indirect rdfs:subClassOf link in the display summarizes the chain of direct rdfs:subClassOf relations, [Process -> Physical -> Entity -> owl:Thing].

In order to help users to quickly get an idea of the size of a particular part of the ontology, for each node displayed, KC-Viz shows two numbers, indicating the number of direct and indirect subclasses. We refer to these as *subtree summaries*. For instance, Figure 1 tells us that class Process has 10 direct subclasses and 879 indirect ones. More information about a class can be found by hovering over the node in question, as shown in Figure 2. Alternatively, if a more thorough analysis of the definition is required, the user can right-click on the node and then select the Inspect menu item, to open up the definition of the class in the Entity Properties View of the NeOn Toolkit.

Once an initial visualization is produced, it is possible to use it as the starting point for a more in-depth exploration of the various parts of the ontology. To this purpose, KC-Viz provides a flexible set of options, allowing the user to control at a rather fine-grained level the extent to which she wishes to open up a particular part of the ontology. For example, let's assume we wish to explore the subtree of class Process in more detail, to get a better understanding of the type of processes covered by the ontology. Figure 3 shows the menu which is displayed, when right-clicking on class Process and selecting Expand. In particular, the following four options (corresponding to the four panes of the window shown in Figure 3) for customizing node expansion are available:


Process [10,879]

Type: Key concept

URI: file:/Users/EnricoMotta/Desktop/NeOn%20Workspace%202/SUMO-LARGE/SUMO.owl#Process

Direct superclasses (actual/shown): 1/0

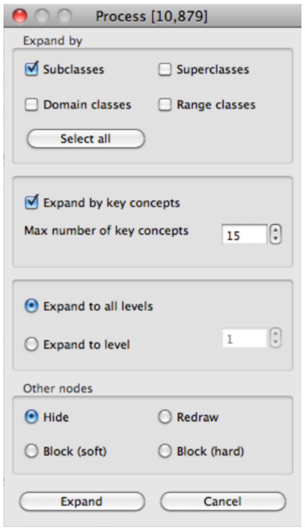
Direct subclasses (actual/shown): 10/2

Individuals (direct/all): 0/7

Figure 2. Tooltips provide additional information about a class.

- Whether to open up the node using taxonomic relations, other relations (through domain and range), or any combination of these. That is, while we primarily use KC-Viz to support sensemaking, focusing on taxonomic relations, KC-Viz can also be used to visualize domain (i.e., non taxonomic) relations.
- Whether or not to make use of the ontology summarization algorithm, which in this case will be applied only to the selected subtree of class Process. As in the case of generating a summary for the entire ontology, the user is given the option to specify the size of the generated summary. Here it is important to emphasize that this option makes it possible to use KC-Viz in a 'traditional' way, by expanding a tree in a piecemeal way, without recourse to key concept extraction. This is especially useful when dealing with small ontologies, or when the user is aware that only a few nodes will be added by the expansion operation, even without recourse to the KCE algorithm.
- Whether or not to limit the range of the expansion – e.g., by expanding only to 1, 2, or 3 levels.

- Whether to display the resulting visualization in a new window ('Hide'), or whether to add the resulting nodes to the current display. In the latter case, some degree of control is given to the user with respect to the redrawing algorithm, by allowing her to decide whether she wants the system to redraw all the nodes in the resulting display (Redraw), or whether to limit the freedom of the graph layout algorithm to rearrange existing nodes (Block Soft, Block Hard). The latter options are particularly useful in those situations where expansion only aims to add a few nodes, and the user does not want the layout to be unnecessarily modified – e.g., because she has already manually rearranged the nodes according to her own preferences. In our view, this feature is especially important to avoid the problems experienced by users with some 'dynamic' visualization systems, where each node selection/expansion/hiding operation causes the system to rearrange the entire layout, thus making it very difficult for a user to retain a consistent mental map of the model.


Process [10,879]

Expand by
[✓] Subclasses [ ] Superclasses
[ ] Domain classes [ ] Range classes

[✓] Expand by key concepts
Max number of key concepts 15

[● Expand to all levels]
○ Expand to level 1

Other nodes
[● Hide] ○ Redraw
○ Block (soft) ○ Block (hard)

Figure 3. Options for further exploration starting from class Process.

The result of expanding the subtree under class Process, using key concepts with the size of the summary set to 15, with no limit to the expansion level, while hiding all other concepts, is shown in Figure 4.

While the flexible expansion mechanism is the key facility provided by KC-Viz to support exploration of ontology trees under close user control, a number of other functionalities are also provided, to ensure a comprehensive visualization and navigation support. These include:

- A flexible set of options for hiding nodes from the display.
- Integration with the core components of the NeOn Toolkit, including the Entity Properties View and Ontology Navigator. This means that it is possible to click on nodes in KC-Viz and highlight them in these components, as well as clicking on items shown in the Ontology Navigator and adding them to the visualization in KC-Viz.

- A dashboard, shown in Figure 5, which allows the user to move back and forth through the history of KC-Viz operations, to modify the formatting of the layout, and to save the current display to a file, among other things.
- A preferences panel, which allows the user to set defaults for the most common operations and also enables her to switch to a more efficient (but sub-optimal) algorithm when dealing with very large ontologies.

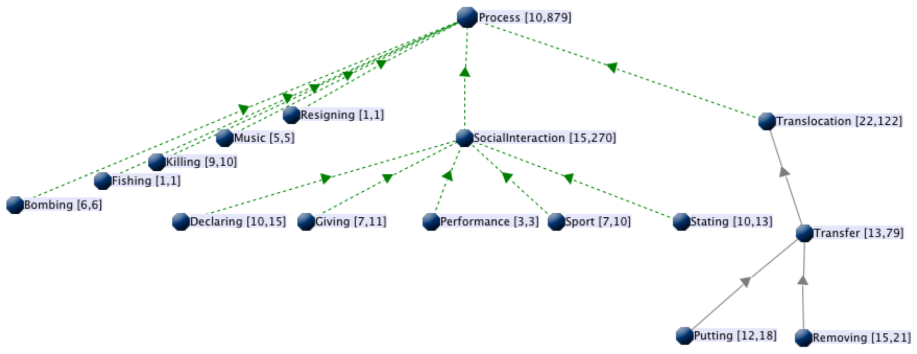


Figure 4. Expanding class Process by key concepts.

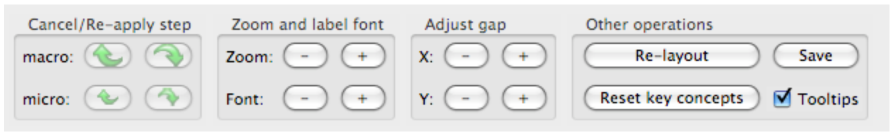


Figure 5. The KC-Viz dashboard.

4 Empirical Evaluation

4.1 Experimental Setup

4.1.1 Tool Configurations

In order to gather initial data about the performance of KC-Viz, we have carried out a preliminary empirical evaluation, which required 21 subjects to perform four ontology engineering tasks, involving ontology exploration. The 21 subjects were drawn from the members of the Knowledge Media Institute, the Computer Science Department at the University of Bologna, and Isoco iLab and were randomly allocated to three different groups, labeled A, B, and C, where each group used a particular configuration of ontology engineering tools.

In particular members of group A carried out the tasks using the NeOn Toolkit v2.5, without any visualization support. More precisely, they were only allowed to use the search functionality, the Ontology Navigator and the Entity Properties View. The role of this group was to provide a baseline to the experiment, providing us with some data on how effectively people can tackle ontology exploration tasks, without any visualization support. The members of Group C were asked to solve the tasks using KC-Viz[7], together with the search functionality provided by the NeOn Toolkit. To ensure a separation between groups A and C, members of the latter group were explicitly forbidden from using the Ontology Navigator for exploration, although they were allowed to use it as an interface between the search facility in the NeOn Toolkit and KC-Viz[8]. Finally, the members of Group B carried out the tasks using the Protégé 4 environment, v4.1.0, in particular using the search functionality, the class browser and the OwlViz plugin. This configuration was chosen for three reasons: i) we wanted to compare KC-Viz to a robust tool, widely used in concrete projects by members of the ontology engineering community[9], to maximize the value of the experiment to the community; ii) while OwlViz uses the same node-link paradigm as KC-Viz, its design is rather different from KC-Viz; and iii) having considered the visualizers available in other state of the art ontology engineering tools, such as the NeOn Toolkit (Kaon Visualizer) and TopBraid (Graph View), we subjectively concluded that OwlViz appears to provide a more user friendly and flexible functionality, than the comparable ones available in TopBraid and the NeOn Toolkit.

4.1.2 Exploration Tasks

For the tasks we used a smaller version of the SUMO ontology, compared to the one referred to in section 4, which comprises 630 classes[10]. SUMO was chosen because, as an upper-level ontology, it is reasonable to expect most people to have familiarity with the notions it covers, in contrast with highly specialized ontologies in technical domains. This particular version of SUMO was chosen for a number of reasons:

- After running a couple of pilots, it became obvious that the ontology provided enough complexity to challenge the subjects and to potentially provide data about the effectiveness of different tool configurations.
- An ontology with thousands, rather than hundreds, of nodes would have required more time for the experiment, potentially reducing the number of subjects willing to take part.
- The more complex the ontology, the higher the risk that a high number of subjects (many of whom could not be considered as experienced ontology engineers) would not complete the task, thus potentially reducing the number of useful data points.

The tasks given to the subjects are shown in Table 1. This set of tasks was designed to ensure coverage of different exploration strategies, which are typically required in the context of a sensemaking activity[11]. Task 1 can be seen as a 'pure'

---

[7] The members of Group C used version 1.3.0 of the KC-Viz plugin, which is part of the core set of plugins included with version 2.5 of the NeOn Toolkit.

[8] The search facility in the NeOn Toolkit locates an entity in the Ontology Navigator and then the user can use the "Visualize in KC-Viz" menu item, to add the located entity to the current KC-Viz display.

[9] To our knowledge Protégé is the most widely used ontology engineering environment currently available.

[10] This can be found at http://www.ontologyportal.org/translations/SUMO.owl.

[11] It is important to emphasize that there is no direct mapping between KC-Viz features and the evaluation tasks chosen for this study. This is not accidental, as we are not interested in performing experiments on tasks which are artificially manufactured for KC-Viz. In addition,

topological task, along the lines of the tasks used in the evaluation described in [6], in the sense that it asks the user to locate a node with a specific topological property. Task 2 is similar to Task 1, however it also requires the user to examine, as a minimum, the labels of the classes, rather than considering them only as abstract nodes in a node-link diagram. Tasks 3 and 4 require a mix of top-down and bottom-up exploration of the ontology and in addition Task 4 requires the user to understand part of the ontology at a deeper level than mere topological structure. Moreover, Task 4 also allowed us to test to what extent tools are able to help when the ontology has a non-standard conceptualization, which may easily confuse users, whether experts or novices. In particular, the SUMO ontology models class CurrencyCoin as a subclass of class Text, which is something many people could find surprising.

T1. Which class has the highest number of direct subclasses in the ontology?
T2. What is the most developed (i.e., has the biggest subtree) subclass of class Quantity found in the ontology at a concrete level of granularity (i.e., do not consider abstract classes which have the term 'quantity' in their id)?
T3. Find three subclasses of Agent, at the most abstract level possible (under Agent of course), which are situated at the same level in the hierarchy as each other, and are also subclasses of CorpuscularObject.
T4. We have two individual entities (a particular copy of the book War&Peace and a particular 5p coin). Find the most specific classes in the ontology, to which they belong, say P1 and P2, and then identify the most specific class in the ontology, say C1, which is a superclass of both P1 and P2 – i.e., the lowest common superclass of both P1 and P2.

Table 1. Ontology Engineering Tasks.

For each task, the subjects were given a 15 minutes time slot. If they were not able to solve a particular task within 15 minutes, that task would be recorded as 'fail'. Before the experiment, every subject filled a questionnaire, answering questions about his/her expertise in ontology engineering, knowledge representation languages, and with various ontology engineering tools, including (but not limited to) NeOn and Protégé. None of the subjects had much direct experience with the SUMO ontology.

4.1.3 Session Setup

At the beginning of the session a subject would be briefed about the purpose of the experiment. To avoid biases in favour or against a particular tool, subjects were simply told that the purpose of the experiment was "to test different configurations of ontology engineering tools". The subject would then be given a tutorial (max 10 minutes) about the specific set of tools he/she would be using. The tutorial was given by the person in charge of the experiment (the 'administrator'). In total four administrators were used. To minimize differences between the tutorials given by different administrators, these were given a precise list of the features that ought to be shown to each specific group. For the tutorial we used the Pizza ontology v1.5, which can be found at http://www.co-ode.org/ontologies/pizza/2007/02/12/.

After the tutorial, the subjects were asked to do a 'warm-up task'. This was exactly the same as T1, however it was carried out on a rather small ontology, AKTLite, a

to ensure repeatability, the evaluation tasks used in this study are rather fine-grained and are associated to precise performance criteria.

subset of the AKT reference ontology[12], which has been used in a variety of projects and applications for representing data about academic organizations. While the AKT ontology contains 170 classes, the AKTLite ontology only contains 85 classes. The AKTLite ontology consists of two sub-ontologies, AKT Support and AKT Portal, and can be found at http://technologies.kmi.open.ac.uk/KC-Viz/evaluation/AKTLite.zip[13]. The subjects were given 10 minutes to solve the warm-up task.

All the tasks, including the warm-up task, were recorded using screen capture software. After completing the task, the subjects were asked to fill a SUS usability questionnaire[14] and to provide qualitative data about their experience with the particular tool configuration they used, including overall impression of the tool, strengths and weaknesses, etc. Finally, the subjects in groups A and B were given a demo of KC-Viz and asked to provide feedback about the tool. This allowed us to get feedback about KC-Viz from all 21 participants in the evaluation.

4.2 Results

4.2.1 Task Performance

Out of 84 tasks in total (4 * 21), 71 were completed within the 15 minutes time limit, while 13 tasks were not completed, a 15.47% percentage failure. The 13 failures were distributed as follows: 5 in group A (NTK), 6 in group B (OwlViz), and 2 in group C (KC-Viz). Table 2 shows the average time taken by each group in each task, as well as the total averages across groups and tasks[15]. As shown in the table, on each of the four tasks the fastest mean performance was with KC-Viz, whose overall mean performance was about 13 minutes faster than OWLViz, which in turn was about two minutes faster than NTK. The mean performance time for OwlViz was faster than NTK for task 3, slower for the others. Although not significant, the difference in total time taken across the four tasks with the three different tools appeared to be approaching significance, F(2, 20) = 2.655, $p = 0.098$.

The difference in performance across the three tools on Task 1, was statistically significant F(2, 20) = 9.568, $p < 0.01$. A Tukey HSD pairwise comparison revealed a significant difference between both KC-Viz and NTK ($p < 0.01$) and KC-Viz and OwlViz ($p < 0.01$), however not between NTK and OwlViz. Although mean performance was faster for KC-Viz across the board, performance differences on the other three tasks did not reach statistical significance. By some margin, the least significant result was found for Task 4 ($p = 0.755$). As discussed earlier, Task 4 involved more problem solving steps than the other tasks (i.e., finding direct parent classes for suggested instances and then their common parent) and an answer that was counter-intuitive to many of the subjects (i.e., a CurrencyCoin being a subclass of

[12] http://www.aktors.org/publications/ontology/.

[13] For the sake of repeatability all tools and all ontologies used in the evaluation are publicly available online, while the tasks carried out in the evaluation are described in this paper.

[14] http://www.usabilitynet.org/trump/documents/Suschapt.doc.

[15] For tasks not completed within the time limit, we consider a 15 minutes performance. This could be modified to consider 'penalties', such as a 5 minutes penalty for a non-completed task. However, adding such penalties does not lead to meaningful changes in the interpretation of the data, other than increasing the performance gap between the KC-Viz group and the others.

Text). Due to the more complex nature of the problem, we hypothesize that other factors, beyond the features provided by a particular ontology engineering tool, influenced performance on this task.

Nevertheless these results suggest advantages for KC-Viz in supporting users in such realistic browsing and visualization tasks. In particular it is reasonable to assume that increasing the sample size beyond the seven per condition in the current study could be expected to lead to statistical significance for overall performance and possibly also for other individual tasks.

	NTK		**OWLViz**		**KCViz**		**Overall**	
	mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.
Task 1	12:03	02:51	12:19	04:16	05:10	03:07	09:50	04:44
Task 2	06:43	04:45	07:20	03:55	04:03	02:15	06:02	03:52
Task 3	11:00	05:31	07:24	04:27	06:25	05:06	08:16	05:12
Task 4	08:01	05:56	08:23	05:28	06:17	05:15	07:34	05:21
Total	37:47	15:02	35:26	15:36	21:55	10:32	31:43	15:01

Table 2. Experimental results (in min:secs).

It is interesting to note that it is the first task that most clearly distinguished the performance of KC-Viz relative to the other tools. This was the first of the four tasks that used the SUMO ontology. Performance on this task would therefore have involved developing some initial overall conceptualization of the ontology, its structure, size and scope, as well as finding ways to navigate it. It is possible therefore that the use of KC-Viz is particularly effective, when users are confronted with large and unfamiliar ontologies.

4.2.2 Other quantitative findings

Usability scores were calculated using the SUS formula for each of the three conditions – see Table 3. The mean usability score was slightly higher for KC-Viz, though very similar across the three tools and not statistically significant.

	NKT		OwlViz		KC-Viz	
	mean	s.d.	mean	s.d.	mean	s.d.
Usability score	26.9	5.1	25.7	4.3	27.1	5.8

Table 3. Usability scores.

However, for each subject, two sub-scores were calculated from the experience questionnaire. The first seven questions in the questionnaire are related to experience with ontologies and ontology languages. The scores on these questions were summed to give a measure of ontology experience. The scores on the final six questions were summed to give a score related to experience with ontology engineering tools.

A positive correlation was found between the ontology experience score and the usability score, $r = 0.546$, $p < 0.05$, while a significant correlation was not found between the score for experience with tools and the usability score. This appears to indicate that perceived usability probably reflects the greater ability of subjects, who are more experienced in the use of ontologies, to adapt to the features, and compensate for the shortcomings, of whatever tool provided for the task. These findings also suggest that the results of usability questionnaires in this kind of evaluations should be treated with much caution and ideally triangulated with other sources of data.

The ontology experience score also had a significant negative correlation with the total time spent across the four tasks (i.e. the higher the ontology experience, the lower the time to complete the task), $r = -0.476$, $p < 0.05$, as well as on task 3, $r = -0.511$, $p < 0.05$. Correlation between experience of ontology engineering tools and task performance was statistically significant for task 1 ($r = -0.469$, $p < 0.5$) and task 3 ($r = -0.452$), and was close to significance on overall performance ($r = -0.410$, $p = 0.065$).

These findings suggest that prior experience with both ontologies and associated tools increases task performance regardless of the toolset used. The deeper understanding that the expert has of the underlying ontological constructs and the heuristics and techniques developed through experience allows the expert to more easily interpret and adapt to whatever tool is provided. Therefore, both differences in performance and usability judgements can be expected to be harder to find when testing with experts than when testing with users with lower levels of experience.

Given the effect of experience on usability judgements and performance, an analysis was conducted to verify that performance differences across the three tools were not due to a skewed distribution of experience across the three conditions. However, experience scores were similar across the subjects in the three conditions and were not statistically significant. This demonstrates that variation in performance across the tools was not due to a bias in the distribution of experience across the three conditions.

4.2.3 Qualitative Results

As already mentioned, the free text questions on the post-task questionnaire elicited views on the perceived strengths and weaknesses of the tool used by each subject. Additionally, subjects who did not use KC-Viz provided feedback following a demo.

A *grounded theory* approach [16] was used to build categories of comments that either expressed positive feedback, offered criticism, or suggested improvements. Categories were discarded when they only contained comments from a single subject. Because of the page limit constraint on this paper, we do not have enough space here to discuss this analysis in detail, hence we only highlight the main findings.

The three main categories of positive comments concerned the flexible support provided by KC-Viz to manipulate the visual displays; the abstraction power enabled by the KCE algorithm; and the value of the subtree summaries provided by KC-Viz. These results are encouraging in the sense that they provide some initial indication that there is probably a direct causal link between the use of key concepts as an abstraction mechanism and the good performance of KC-Viz on the evaluation tasks, even though these were not designed specifically to map directly to KC-Viz features.

The three main categories of negative comments included: criticism of the tree layout algorithm used by KC-Viz, which does not allow display rotation and at times generates overlapping labels; the lack of transparency of the KCE algorithm, which does not allow the user to configure it, or to clarify why a node is considered more important than others; and the lack of integration between KC-Viz and reasoning/query support in the NeOn Toolkit.

5 Conclusions

Exploring a large ontology, particularly when it is unfamiliar to the user, can be characterized as *information foraging* [17]. Information foraging theory, drawing on ecological models of how animals hunt for food, proposes the notion of *information scent*. An animal foraging for food will follow a scent in order to locate a promising patch rich in sources of food. Analogously, an information forager will follow an information scent in order to locate rich information sources. In a hypertext environment, the name of a link, a preview of the information source, or even the source URL may give the information forager clues as to the potential profitability of following the link. This helps the forager to choose between alternative paths in the search for information.

The support provided by KC-Viz for displaying key concepts and using these as the basis for further exploration can be seen as assisting information foraging from an ontology. In particular, the flexible set of options provided by KC-Viz for manipulating the visualization enables the user to construct a view on the ontology that allows them to compare the information scent of different paths through the ontology and control how they pursue these paths. Key concepts use a number of factors to estimate the importance of a particular class and therefore provide means for estimating the potential profitability of an information scent. In addition, subtree summaries provide a topological clue as to the potential profitability of a class.

This perspective might help explain why KC-Viz was found to be particularly advantageous when exploring a large ontology for the first time and indeed both key concepts and subtree summaries were highlighted as strengths of the approach, while the lack of these kinds of abstraction/summarization mechanisms were identified by users as a deficit of both the NTK and OwlViz configurations.

The empirical study also offers lessons learned for future evaluations of ontology browsing and visualization tools. In particular it showed that significant prior experience in ontology engineering enables users to adapt well to different tool configurations and perform well regardless of the specific configuration they are using. In addition, it also showed that experts can have relatively positive usability judgments of interfaces. Both of these observations suggest that users having a broad range of expertise should be gathered for usability testing and that the results from experiments that fail to triangulate multiple sources of data, including usability scores, task performance and qualitative feedback, should be treated with caution.

Our future work has two broad strands. First of all, we intend to further develop KC-Viz, taking on board the feedback gathered during the evaluation. In particular, improving the layout algorithm, opening up the KCE algorithm to users, and integrating KC-Viz with ontology reasoning and querying are all priorities for development. In addition, we also intend to capitalize on the 21 videos collected during the evaluation, both to undertake a fine-grained analysis of the navigational strategies employed by users, and also to uncover possible misconceptions revealed in the use of the various ontology tools. It is hoped that from these analyses we will then be able to generate additional design recommendations for future versions of KC-Viz.

References

1. Oren, E., Delbru, R., Catasta, M., Cyganiak, R., Stenzhorn, H. and Tummarello, G. (2008). Sindice.com: a document-oriented lookup index for open linked data. Int. J. Metadata, Semantics and Ontologies, Vol. 3, No. 1, pp.37–52 (2008).
2. d’Aquin, M. and Motta, E. (2011). Watson, more than a Semantic Web search engine. Semantic Web 2(1), IOS Press.
3. Noy, N. F., Tudorache, T., de Coronado, Sh., and Musen, M. A. (2008). Developing Biomedical Ontologies Collaboratively. In Proceedings of the AMIA Annual Symposium, pp. 520–524.
4. Dzbor, M., Motta, E., Buil Aranda, C., Gomez-Perez, J.M., Goerlitz, O., and Lewen, H. (2006). Developing ontologies in OWL: An observational study. Workshop on OWL: Experiences and Directions, November 2006, Georgia, US.
5. Peroni, S., Motta, E., d'Aquin, M. (2008). Identifying key concepts in an ontology through the integration of cognitive principles with statistical and topological measures. Third Asian Semantic Web Conference, Bangkok, Thailand.
6. Wang, T.D. and Parsia, B. (2006). Cropcircles: Topology Sensitive Visualization of Owl Class Hierarchies. In Proceedings of the 5th International Semantic Web Conference, Georgia, US.
7. Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In Proceedings of the 1996 IEEE Symposium on Visual Languages (VL '96). IEEE Computer Society, Washington, DC, USA.
8. Shneiderman, B. (1992). Tree Visualization with Tree-Maps: A 2d Space-Filling Approach. ACM Trans. Graph., 1992. 11(1): p. 92-99. 15.
9. Plaisant, C., Grosjean, J., and Bederson, B. B. (2002). Spacetree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. In Proc. of the Intl. Symposium on Information Visualization.
10. Storey, M. A., Musen, M.A., Silva, J., Best, C., Ernst, N., Fergerson, R. and Noy, N.F. (2001). Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protege. Workshop on Interactive Tools for Knowledge Capture, K-CAP-2001, Victoria, B.C., Canada.
11. Kriglstein, S. and Motschnig-Pitrik, R. (2008). Knoocks: A New Visualization Approach for Ontologies. In Proceedings of the 12th International Conference on Information Visualisation (IV '08), pp. 163-168. IEEE Computer Society, Washington, DC, USA.
12. Katifori, A., Halatsis, C., Lepouras, G., Vassilakis, C., and Giannopoulou, E. (2007). Ontology Visualization Methods - a Survey. ACM Computing Surveys, 39(4).
13. Souza, K., Dos Santos, A., et al. (2003). Visualization of Ontologies through Hypertrees. In Proc. of the Latin American Conference on Human-Computer Interaction. pp. 251-255.
14. Katifori, A., Torou, E., Halatsis, C., Lepouras, G., Vassilakis, C. (2006). A Comparative Study of Four Ontology Visualization Techniques in Protege: Experiment Setup and Preliminary Results. In Proceedings of the 10th Int. Conference on Information Visualisation (IV'06), pp. 417-423, London, UK.
15. Rosch, E. (1978). Principles of Categorization. Cognition and Categorization. Lawrence Erlbaum, Hillsdale, New Jersey (1978).
16. Birks, M., and Mills, J. (2011). Grounded Theory: A Practical Guide. SAGE Publications Ltd.
17. Pirolli, P. and Card, S. K. (1999). Information Foraging. Psychological Review, 106 (4), pp. 643-675.