

Cognitive Support for Ontology Modeling

Neil A. Ernst Margaret-Anne Storey Polly Allen

*Department of Computer Science
University of Victoria
PO Box 3055, STN CSC
Victoria, BC, Canada V8W 3P6*

Abstract

Knowledge engineering tools are becoming ever more complex, and therefore increased cognitive support will be necessary to leverage the potential of those tools. Our paper motivates this claim by examining some previous work in this domain and explaining the nature of cognitive support. We discuss some of the problem areas we have encountered in our research. Through user questionnaires and observations carried out at the National Cancer Institute (NCI) and the University of Washington Foundational Model of Anatomy (FMA) Project, we have begun to gain an understanding of the cognitive barriers experienced by the users of knowledge modelling tools. We analyze some existing solutions developed for the Protégé knowledge engineering tool using this understanding, one of which is our own tool, called Jambalaya. We then analyze the support Jambalaya provides using some non-functional design criteria and illustrate some trade-offs inherent in tool design. We suggest that the need for cognitive support in knowledge engineering is immediate and essential.

Key words: Cognitive support, Knowledge engineering, Information visualization, Requirements engineering

Email addresses: nernst@uvic.ca (Neil A. Ernst), mstorey@uvic.ca (Margaret-Anne Storey), allenp@uvic.ca (Polly Allen).

1 Introduction

This paper discusses the need for increasing the cognitive support knowledge engineering tools provide. Cognitive support leverages innate human abilities, such as visual information processing, to improve human understanding and cognition of challenging problems (Walenstein, 2002a). For example, most of us require a visual and auditory indicator that our turn signal is on, since human limitations in short-term memory restrict our ability to recollect this information after a few seconds.

While the cognitive support offered by knowledge engineering tools of old was acceptable, an increased adoption rate of those tools means knowledge engineering will now require tools which are not only logically rigorous, but also supportive of user understanding of those tools and their products. This is due to several factors. For one, the applications of knowledge engineering are growing larger and more systematic, now encompassing much larger ontologies – sizes in the hundreds of thousands of concepts will not be uncommon. Furthermore, new technologies proposed by the World Wide Web Consortium’s semantic web initiative promise to bring the benefits (and the drawbacks) of knowledge engineering to many more people. We believe this change in focus brings with it an associated need for enhanced cognitive support.

Our paper motivates this claim by examining some previous work in this domain and explaining the nature of cognitive support. We then discuss some of the problem areas we have encountered in our research, and present some of the approaches to those problems, including our tool. We then analyze the support Jambalaya provides according to non-functional design goals, and illustrate some trade-offs inherent in tool design. We conclude with a brief look at one way to increase cognitive support in these tools, customization.

2 The need for cognitive support

The field of knowledge capture and knowledge representation is at a crossroads. The preceding years have been characterized by *ad-hoc* development, and lately there has been a move towards more of a systematic approach to the development of knowledge-based systems. In other words, “[t]his requires the analysis of the building and maintenance process itself and the development of appropriate methods, languages, and tools specialized for developing [knowledge-based systems] (Studer *et al.* (1998), p. 1).” In a survey we conducted, detailed in section 5.2.1, we found more than half of knowledge engineering projects involve five or fewer individuals. In terms of the increase in reliability and size of applications, knowledge engineering is largely dominated by the hobbyist and the researcher. However, demand

for these applications is causing many to grow in scope and scale. Such growth in turn demands adoption of more systematic processes of development; a trajectory which closely mirrors that of software development. In addition, the successes of the world wide web have shown many people the power of distributed information – who doesn't know the term 'google'? – and led some researchers and developers to leverage knowledge engineering techniques to provide even more meaningful information - such as those the semantic web initiative Berners-Lee *et al.* (2001) describes. This vision foresees many different ontologies used to describe data and domains, enable interoperability, and secure transaction success, and as such will have a large impact on the knowledge engineering domain. We believe that one of the byproducts of this success will be a requirement for much improved metaphors for understanding the ontological commitment that these different ontologies make. Numerous different taxonomies, vocabularies, and data dictionaries will place a heavy cognitive burden on knowledge engineers and domain experts as these individuals attempt to comprehend the models these structures make. While logical formalisms certainly have their place in knowledge modeling, other researchers have shown that formal, logic-based language is difficult for humans to comprehend. For example, Shipman and Marshall make this point in their paper “Formality Considered Harmful” (Shipman and Marshall, 1999).

Cognitive challenges are by no means new to this field. Indeed, a book edited by James Hendler (Hendler, 1988) – one of the leading proponents of the semantic web – mentions some of these issues and challenges with the user interfaces of expert systems. Stelzner and Williams (1988), in particular, describe some of the concerns with mapping between the user's mental model of a system, and the system model itself. In a tool designed to help students understand a medical expert system related to MYCIN, called GUIDON-WATCH, Richer and Clancey (Richer and Clancey, 1985) make the point that “. . . providing multiple views of the same knowledge or behavior can help a user understand a complex system”. From these very early studies, it seems apparent that new systems are going to require well-designed interfaces to make the semantic web vision at all feasible. Well-designed implies that there has been some form of iterative, user-centered design methodology involved in creating the system. For instance, a tool may leverage techniques and tools from *information visualization* to create advanced visual interfaces. Information visualization tools are specifically designed to help humans understand and navigate complex information spaces.

Cognitive support in this context refers to using such a user-centred design process to analyze what aid the tool can provide in the decision-making and understanding process (see Walenstein (2002b) for a discussion of this). Developers should consciously question how they are going to support the increasingly complicated tasks faced by knowledge engineers and modelers. For example, does the tool support understanding the ontological commitment a model makes?

Lack of complete understanding of cognitive support issues for knowledge engi-

neering tools partly explains the lack of adoption. This is particularly relevant in knowledge engineering since the cognitive burdens on users will only increase. For example, in one typical example the size of the domain model, let alone the domain itself, is larger than what any one user can comprehend. This model, known as the National Cancer Institute Thesaurus (NCI Thesaurus), “facilitates the standardization of vocabulary across the Institute and the larger cancer biomedical domain” (see <http://ncicb.nci.nih.gov/core/EVS>), containing “detailed semantic relationships among genes, diseases, drugs and chemicals, anatomy, organisms, and proteins Golbeck *et al.* (2003)”. Providing cognitive support at appropriate places will be of great assistance to modelers. We discuss some of these areas in more detail later on. We also discuss some non-functional design goals tool developers need to consider.

3 Background

Before presenting related work in this area, there are a few aspects covered in this paper that are worthy of introductory explanations for clarification.

3.1 User types

It is possible to outline many different user types for ontology engineering. For example, one might consider the actors to fill the roles of development manager, lead modeler, domain expert, expert user, and end user. Such a taxonomy is non-exhaustive and limiting. We prefer to abstract from the specifics, instead noting that while there are many actors involved in the development and use of an ontology, typically there are two major roles played (any of which may belong to different people at different stages). We characterize these as developers or modelers, and end-users. For example, from our work at the National Cancer Institute, the modeler role is played by the scientists who are concerned with the knowledge acquisition process, while the end user role is played by scientists who leverage the NCI Thesaurus at their own centres (for example, to provide terms for querying and report creation).

In general, this paper is focused on the modeler role. This implies that our findings below are targeted to this role; while end users and novices may have different requirements with which we are not yet completely familiar, this category of user is certainly important, and our lack of attention should not be construed as not recognizing its significance.

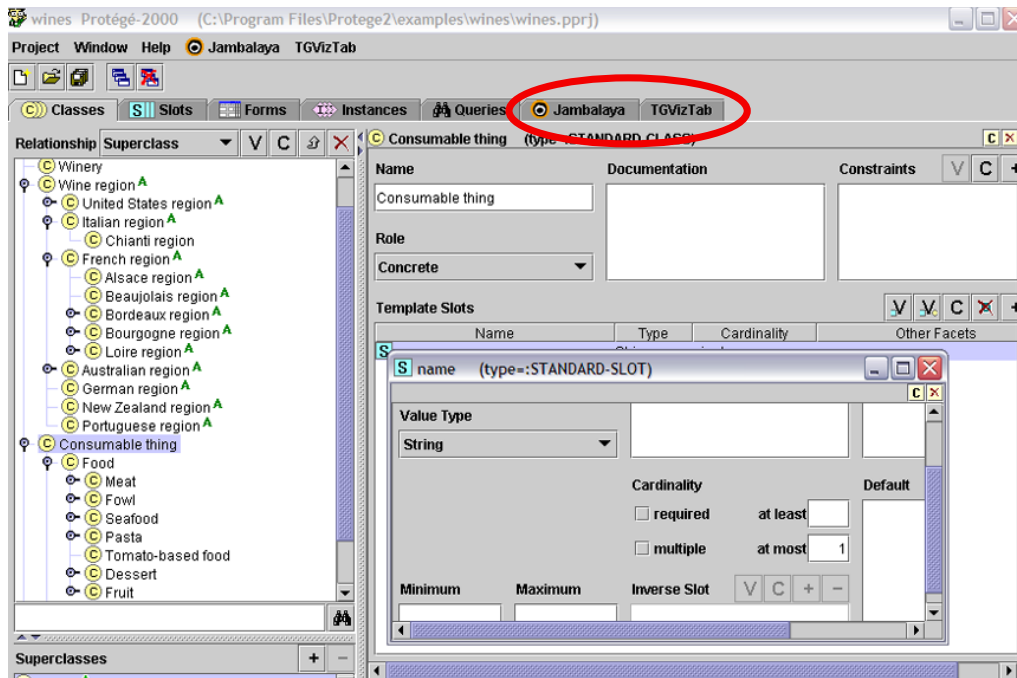


Fig. 1. The Protégé user interface with the Jambalaya and TGViz tabs visible

3.2 Information Visualization

Information visualization techniques are used in many domains to help provide insight or to communicate information (Card *et al.*, 1999). Information visualization leverages innate human abilities to perform spatial reasoning and make sense of relatively complex data using some form of graphical representation language. In the domain of knowledge engineering, such a language is often based on graph theory and has two components: one, the use of nodes to represent concepts in a domain; the other, the use of edges to represent relationships between concepts. The language for visualizing information in this domain therefore consists of manipulations of graphs in some form or another. Information visualization is a technique for constructing advanced visual interfaces.

3.3 Protégé

Protégé (Gennari *et al.*, 2003) is an ontology engineering and knowledge acquisition tool created at Stanford University. It uses a frame-based knowledge representation formalism to allow users to model domains using classes, instance, slots (relations), and facets (constraints on the slots). Written in Java, its architecture allows for extensions to be added via a plug-in metaphor (as shown in Fig. 1). Recently, work has been ongoing to make the tool compatible with the OWL ontology language for the Semantic Web (a key component), as well as support web-specific

concepts such as namespaces and Universal Resource Indicators (URIs). More detail is available in Knublauch *et al.* (2003).

4 Related work

As mentioned in the introduction, there has been some research into user support for knowledge engineering, largely in the area of user interfaces for expert systems. This work was motivated in large part by the realization that few users could easily understand what a tool was doing or how to make it work. Some early knowledge representations were directly graphical, such as Sowa's work on conceptual graphs (Sowa, 2000). This representation format, while certainly more readable than first-order logic based representations such as KIF (Ginsberg, 1991), focused more on being logically rigorous than providing cognitive support for end-users. A similar tool, concept maps (Gaines and Shaw, 1995), were more focused on user support, but seemed to lack rigorous logical representations for knowledge acquisition and inferencing. CYC, an effort to represent common-sense knowledge, had several user interfaces built for it, of which one, by Travers, used a room metaphor to browse different areas of the ontology (Travers, 1989). However, this metaphor has difficulty with different relationships. Another knowledge representation tool, CODE4 (Skuce and Lethbridge, 1995), focused in more detail on the user experience, and also combined it with a logically rigorous representational semantics. A key detail that CODE4 focused on was providing multiple methods to view the knowledge, emphasizing the separation of presentation from model. For example, the system provided graph layouts of the knowledge base, but also provided a tabular interface.

Other early work that is applicable to this subject includes the research done in visual programming, particularly in Expert Systems. Visual programming is important because the tasks associated with it (program understanding, control flow, model checking) are highly consistent with ontology engineering tasks, as we shall see in more detail. A good example of such a system is KEATS, a knowledge engineering tool with strong emphasis on visual programming and program visualization (Eisenstadt *et al.*, 1990). KEATS supported the notion of sketching early versions of a knowledge base before the actual design commenced. This differs slightly from the focus in this paper, which is more concerned with how modelers understand or verify a model after it has been (largely) completed.

The GKB-Editor tool (Karp *et al.*, 1999) has a graphical interface for visualizing and editing a frame-based knowledge base. It has several views, such as a hierarchical view of the concept hierarchy, a relationships viewer, and a spreadsheet viewer. However, the views are static once defined, and do not allow much customization and interaction on the part of the user.

Another set of tools applied visualization techniques in information retrieval and management. An early work, SemNet (Fairchild *et al.*, 1988), had several complex metaphors for visualizing personal information, including fisheye views, zooming, and hyperlinking; however, the hardware available at the time (1988) greatly limited its adoption, as did the relatively small amount of electronic data. Other work built on the graph visualization theme, discovering new techniques for browsing networked data. A lot of work has been done on visualizing hypertext networks (closely related to concept maps). For example, VISAR (Clitherow *et al.*, 1989) was an early graphical tool to aid in the construction of hypertext systems, again using CYC.

5 Determining where cognitive support can help

In order to discover more details about where we believed cognitive support might help, we used a number of research techniques, including background research, surveys, and qualitative evaluations. We introduce the research by describing some of the reasons we felt compelled to investigate this subject.

5.1 *Impetus for the research*

Over the past few years, our research group at the University of Victoria has developed a tool similar to those mentioned in the background review. Jambalaya – available at shrimpviews.org – was initially developed as the result of a collaboration between the Protégé team and Jambalaya developers (Storey *et al.*, 2001). Jambalaya was based on a tool for visualizing software systems, and we recognized some obvious synergies with knowledge engineering. This initial development and release was characterized by a lack of any formal requirements process. While it was apparent to both teams that this tool was potentially very useful, insufficient effort was made to identify what the tool should look like; thus, the existing features were simply inserted wholesale into Protégé. We had identified requirements for the tool based on our work in software comprehension (Storey *et al.*, 1999), but had not done any work to identify the requirements in this different domain.

Approximately six months following the initial release of the plug-in, we began to question whether the tool was meeting the requirements of the users. We did limited quantitative analysis of the Jambalaya tool: we conducted some studies to analyze how useful the various interface elements were, but nothing detailed was done from a quantitative perspective. A heuristic evaluation was also done to identify some user interface challenges, which helped to make the tool more usable. Neither approach, however, provided any clear requirements that our tool should meet.

5.2 *Requirements gathering*

An examination of other visualization tools for knowledge engineering, and the requirements they were built to satisfy, revealed the lack of an established theory about user tasks and the cognitive support they require. This examination is discussed in more detail in one of our papers, Allen's thesis (Allen, 2003). Without this theoretical guidance, quantitative approaches – such as formal user testing – would fail to reveal new requirements for such tools. Furthermore, Allen identifies many difficulties encountered when performing user testing in the knowledge

engineering domain, including gaining access to expert users, generalizing results over different domains, and quantifying the knowledge acquired and used by such tools. These issues led us to focus on more qualitative approaches which included a user survey, two contextual inquiries, and investigation of related work; using these different techniques provided a series of useful perspectives on the problem.

5.2.1 *User Survey*

One of our approaches was to survey the general user population of the Protégé tool and determine some of their preliminary needs for a visualization tool. This resulted in a user survey which was disseminated to the Protégé user list and other knowledge engineering lists. The survey is available at <http://shrimp.cs.uvic.ca/jambalaya/user-survey.htm>. The results are discussed in more detail in a report by Ernst and Storey (2003). In general, we found that there are a wide variety of users and domains to which ontology engineering is being applied, and further, that visualization *is* a desired feature for the respondents. The lesson for those working with tools which manipulate and create ontologies is that this diversity must be supported. We believe that the wide-ranging degree of domains we uncovered is a sign of the future, and that tools that operate at a meta level to assist users to understand the modeling decisions, such as Jambalaya, will be increasingly important in maintaining clear communication and understanding.

5.2.2 *Contextual Inquiries*

We used the survey as a pointer to areas where more detailed investigation might be useful, and this drove the second aspect of our requirements gathering: contextual inquiries at two separate venues. The American National Cancer Institute works on a large-scale taxonomy and ontology of cancer-related research and disease concepts, known as the NCI Thesaurus (Golbeck *et al.*, 2003), using a tool suite named Apelon (www.apelon.com). We conducted two site visits to the NCI team to determine how their ontology engineering workflow proceeded, and what some requirements for that workflow might be. Using contextual inquiry techniques (Beyer and Holtzblatt, 1998), we sat alongside users to view their daily activities. We also conducted higher level discussions with the technical gurus and managers of the project, to gauge their needs. The inquiry demonstrated a need for a number of mechanisms. One user demonstrated a reporting tool, custom built, which showed an indented text-based layout of the concept hierarchy and the associated relationships, designed to show “what this branch [of the ontology] looks like”. Modelers also relied extensively on text searching for navigation, using this mechanism to jump around the concepts. They also wanted to define concepts using existing concept definitions (i.e., make a copy and edit that), but this was not supported. Another user used a third-party tool to model the concept textually, and then added the concept using that external information. Finally, all users expressed an interest in more

collaborative, real-time modeling work.

We also conducted a site visit to the University of Washington Foundational Model of Anatomy Project (FMA; see <http://sig.biostr.washington.edu/projects/fm/AboutFM.html>). Here, we employed similar techniques to gauge the needs of the users of that domain. While watching the users perform domain modeling and verification with Protégé-2000, we asked specific questions about the process, and concluded with a demonstration of Jambalaya in its current form to gauge their reaction. From this demonstration we gathered several important points of feedback on how the tool could be changed to better suit their needs. For example, users demonstrated that knowledge engineers would like to visualize not only the taxonomy hierarchy but the meta-data hierarchy as well. Other users identified more support for editing being desirable, such as the ability to ‘drag and drop’ concepts of interest onto a list on the side of the screen. It was observed here as well that modelers spent a lot of time performing text searching operations. Finally, the modelers noted that errors in modeling were found about once every two weeks, indicating that existing verification mechanisms were insufficient.

These two visits, in conjunction with the survey, as well as our previous work in software engineering (Storey *et al.*, 1997), gave us some specific data to form preliminary visualization requirements for the knowledge engineering domain, in conjunction with a detailed literature review, discussed next.

5.3 Related research

We conducted an extensive background review looking for discussions of user challenges in knowledge engineering. Four studies we discovered during our literature review were highly relevant and we discuss them here. Other studies concerned user interface details, but often did not suggest specific areas of concern or problems users encountered. We present these studies in chronological order.

During user studies of a Knowledge Acquisition system conducted by Tallis *et al.* (1999) the experimenters observed users performing the following high-level tasks:

- Understanding the given Knowledge Acquisition task
- Deciding how to proceed with the Knowledge Acquisition task
- Browsing the Knowledge Base to understand it
- Browsing the Knowledge Base to find something
- Editing (create or modify) a Knowledge Base element
- Checking that a modification had the expected effects on the Knowledge Base
- Looking around for possible errors
- Understanding and deciding how to fix an error
- Recovering from an error by undoing previous steps (to delete or restore a Knowledge Base element)

- Reasoning about the system

This study serves to detail some of the tasks that users go through while engaged in the traditional knowledge engineering process. It also hints at some of the problems that an otherwise perfectly valid knowledge engineering project may encounter if it fails to address the specific cognitive needs of the users. For example, a system which failed to provide simple, usable methods for adding knowledge or looking for errors would be quickly rejected by users without a profound investment in the system (typically these would be the actual designers of the system).

Blythe *et al.* (2001) further identified some typical concerns that users may have when adding new knowledge to an intelligent system. Some of these concerns were that the users do not know where to start and where to go next, the users do not know if they are adding the right things, and the users often get lost as it takes several steps to add new knowledge. This study is interesting because it clearly shows that the standard knowledge engineering methodology, consisting of the steps of modeling, acquiring, and verifying knowledge, fails to accommodate the specific needs of users, even modelers in the domain. It is no use to have a crisp and detailed methodology if users cannot easily make use of it in any one of its stages.

Gary Kwok-Chu Ng conducted user studies as part of his Ph.D research (Ng, 2000). Based on an evaluation of user requirements in ontology modeling tasks, he designed a tool, InfoLens, to browse description logic ontologies, using a combination of ‘lenses’ which revealed different information about the domain as they were interactively moved about the model representation. He found that one issue was scalability for practical sized systems. For cognitive support specific tasks, he identified the need for a tool to support information integration (between different representations), to support the often cyclic task-switching between navigation (around the model) and visualization (of a specific aspect of the model). Initial user surveys of his tool were quite positive but some aspects of the implementation hindered the evaluation. Despite these problems, Ng reports his tool was “effective in finding patterns useful for modellers, with a few limitations in both scalability, flexibility and completeness (Ng (2000), p. 212)”. We analyze why these limitations occurred in Section 8. Ng also identifies the same high-level tasks that we outline below in Section 6, although we use the term ‘modeling’ rather than his term ‘authoring’.

Finally, Clark *et al.* (2001) conducted studies of SHAKEN; a graphical tool for knowledge acquisition. Although they only tested it on four users, and those users were not modelers or knowledge engineers, we still present the results for the insight it offers into the benefit of increased cognitive support. The users were able to enter a few hundred concepts into a large medical knowledge base within a week, and also verify the model using competency questions. From the discussion:

[the results suggest] the basic machinery works, providing a basic vehicle for axiom-building without the users having to encode axioms directly (or even en-

counter terms like concept, relation, instance, quantification, etc.) ...

Some of the issues Clark *et al.* identify as areas needing improvement include multi-faceted representations, active critiques from the system, and more expressivity in the interface (such as temporal relations and conditions).

6 Areas Requiring Cognitive Support

This section provides some of the motivation for functional requirements of knowledge engineering tools. Functional requirements refer to specific knowledge engineering tasks – such as knowledge acquisition, modeling, verification, and usage – that cognitive support tools should address. We prefer to term these ‘tasks requiring cognitive support’, rather than functional requirements, as the term requirements traditionally refers to domain and tool specific needs, whereas what we term tasks address high-level commonalities we have seen over a range of knowledge engineering projects. There are also other, non-functional requirements (also known as effectiveness criteria, quality attributes, and constraints) (Mylopoulos *et al.*, 1992), which we discuss in Section 8.

Our list of tasks where we think cognitive support will be of most assistance is based on four streams of research: 1) the detailed literature review of user interfaces to knowledge engineering tools and papers on the few user studies that are available; 2) our own qualitative analyses from site visits to two large and well-known knowledge engineering efforts; 3) our background in software program comprehension, as many researchers have made the case that software engineering and knowledge engineering have many parallels (such as Eisenstadt *et al.* (1990), Devedzic (2002), Falbo *et al.* (2002), Kalfoglou (2000)); and finally, 4) our survey of ontology users identifying visualization requirements and domain specific issues (Ernst and Storey, 2003).

We have made these considerations as specific as possible. What follows is a non-exhaustive list of tasks in the knowledge engineering process we have identified as requiring cognitive support; non-exhaustive, because they are based on our experiences and research, and could change depending on a variety of factors, including the domain of interest and user characteristics. Our objective is to present a fairly detailed list of problem areas that have *emerged* through our work over the past few years. Furthermore, this list is tailored, as mentioned above, to modelers working on the ontology. Clearly, some of these may seem obvious, but we wish to make them explicit. We indicate from which research method these were developed using the following code: [s] for survey, [r] for review, [pc] for program comprehension work, and [ci] for the contextual inquiries. Table 1, following this section, summarizes the findings presented below, and gives specific details on the origin of the requirements mentioned. We highlight three encompassing terms to structure

our taxonomy: navigation, modeling, and verification. These terms are intended to organize the tasks into different sub-areas in the knowledge engineering process, and are not exclusive. For example, navigation tasks may be performed during a verification exercise.

- **Navigation** – support the navigation of ontologies for understanding, discovery, and search
 - (1) *Provide overviews and support top-down exploration of the ontology ([s] [r] [pc] [ci])* – top-down exploration is most useful when people are unfamiliar with the ontology. In large ontologies, in particular, this strategy could be highly useful, because complex ontologies rapidly overwhelm the abilities of users to make sense of where they are, and where they need to be. By providing an overview, a cognitive aid can address some of these challenges. Often, top-down exploration is goal-directed and based on a hypothesis. There is evidence, from the software engineering literature, that navigation strategies often alternate between top-down and bottom-up (Storey *et al.*, 1997), and tools should support them. In the case of modelers, this may occur when working with ontology merging, but we have found no specific evidence to that effect.
 - (2) *Support slot-based browsing ([r] [ci])* – Navigation should allow following hierarchical and network structures. In OWL, for example, slots may be composed of other properties and classes using set-theoretic semantics such as Union and Intersection. For example, one may want to see subclasses of Wine, and then navigate to the regions those wines are produced in. The tool would provide mechanisms to discover relationships that are not clear from the standard view of the model. Furthermore, a tool should show the user values for all slots.
 - (3) *Allow users to view query results ([r] [pc] [ci])* – in some cases, it is sufficient to highlight the concept, but in other cases it may be important to show the user what context the result is in, and perhaps why certain results were ranked lower – to allow users to determine what they ‘meant’ by a search.
 - (4) *Provide a mechanism for saving, annotating, and sharing views ([s] [r] [ci])* – many times we noticed users wanting to share their recent work to show others or save intermediate results. For example, it was noticed that many users wished for a way to save intermediate modeling in the event that they were forced to abandon it. At the NCI, since the elements in the domain of discourse (cancer genetics) change so frequently, it was not uncommon for the modeling staff to restructure entire subtrees. The modeler with chief responsibility for a particular sub-ontology expressed a desire to share his intermediate efforts with the other experts.
- **Modeling** – support modelers in their tasks of constructing the knowledge base or ontology.
 - (5) *Provide graphical editing techniques ([r] [pc])* – allow users to directly edit classes in an ontology. The tool should maintain the user’s context, allowing them to see how the concept being edited relates to neighbours.
 - (6) *Editing navigation ([ci])* – support for “jumping” between two concepts as they

are being joined by a relationship during knowledge modeling. During observations of the FMA team, modelers were continually visiting several detailed parts of a large ontology which were difficult to navigate between. For example, the `Lung` and `Chest Cavity` concepts were in very different areas of the `is-a` hierarchy, but needed to be joined directly by the `part-of` relationship. Creating and verifying that relationship involved using text searching to switch between the two concepts.

- (7) *Support ontology reuse* ([s] [r] [pc] [ci]) – allow users to quickly and easily identify external ontologies and knowledge bases that may be useful, and directly include these in the model; furthermore, the namespace model that many representations are using, based on the World Wide Web Consortium specification, should be clear and explicit.
- **Verification** – support for the visual checking of a knowledge base’s structure, such as the model-model and model-domain consistency.
- (8) *Identify incoming relationships* ([pc] [ci]) – the tool should provide support for the identification of a concept’s incoming relationships, or the lack thereof, during model verification. It is difficult to identify such concepts that depend on other concepts without doing text searches and navigating to each of the results individually. This can be quite cumbersome if the interface does not have specific support for such a task. For example, a modeler may wish to know what concepts link to the one being examined, to determine what effect changes will have. In frame-based systems, it is not difficult to detect outgoing relationships as they are explicitly defined as part of the concept definition.
- (9) *Incremental navigation* ([r] [ci]) – provide support for incremental navigation of a knowledge base. A tool providing this would be able to show the modeler all nodes which are N slots out from the central concept. Here, the concept is clearly of interest, and the user wishes to start investigating from there.
- (10) *Browse multiple and complex relationships at once* ([ci]) – the tool should provide: complex relationship support, including support for understanding a concept’s place in multiple relationships simultaneously, to help verify newly-modeled concepts; and support abstraction of reified relationships. For instance, the FMA modelers reify the `part-of` relationship based on the ‘type’ of a concept, such as an anatomical or physical part of the body.

Category	Task #	Program Compr.	Contextual Inq.	Readings	Survey
Navigation	1	x	x (NCI, FMA)	x ((Clark <i>et al.</i> , 2001), (Tallis <i>et al.</i> , 1999), (Blythe <i>et al.</i> , 2001), (Ng, 2000))	x
	2	-	x (NCI, FMA)	x ((Tallis <i>et al.</i> , 1999),(Blythe <i>et al.</i> , 2001), (Ng, 2000))	-
	3	x	x (FMA)	x ((Blythe <i>et al.</i> , 2001))	-
	4	-	-	x ((Clark <i>et al.</i> , 2001), (Ng, 2000))	x
Modeling	5	x	-	-	x
	6	-	x (FMA)	-	-
	7	x	x (NCI, FMA)	x ((Clark <i>et al.</i> , 2001), (Blythe <i>et al.</i> , 2001))	x
Verification	8	x	x(NCI, FMA)	-	-
	9	-	x (FMA)	x ((Ng, 2000))	-
	10	-	x (FMA)	-	-

Table 1

Research methods used to derive cognitive support areas. An x indicates the area was determined from that research method, and parentheses indicate the origin (literature reference or project).

7 Approaches to cognitive support in knowledge engineering

Many tools already support some of the areas we have identified. Protégé itself, for example, provides extensive cognitive support for navigation, modeling, and verification tasks, otherwise it would not be as successful as it is. In this paper, we will examine tools which provide cognitive support using information visualization to construct an advanced user interface (as well as using the core Protégé as a baseline reference). We do this because examining all features would be well beyond the scope of this article, and because as information visualization researchers, we have a great deal of expertise in this domain. We begin by discussing knowledge engineering tools in general, in order to illustrate the depth of work done in this area. We then turn our focus to Protégé-specific tools, in order to provide detailed examples of how the areas listed in Section 6 are supported.

7.1 Visualization tools outside of Protégé

7.1.1 Spectacle

Spectacle (Fluit *et al.*, 2002) is an ontology-based information visualization solution for large datasets. It creates an *Exploration Context* out of user-supplied structured information, which then groups the data into visual structures, such as job categories and job descriptions. A spring-based cluster map presents an attractive presentation of the data and interconnections between data elements. However, the ontologies used in Spectacle are lightweight taxonomies, and it supports primarily navigation and querying operations of instances, and not the model itself.

7.1.2 IsaViz

Isaviz is a tool designed by the World Wide Web Consortium (W3C) to visualize knowledge representations constructed using the Resource Description Framework (RDF). It uses the GraphViz library from AT&T. Although the user can configure how the views appear, they are not very interactive nor easily customized. The parsing and generating of graphs can be quite slow. It also has facilities for styling the graph using a stylesheet concept, exporting to the popular Scalable Vector Graphics (SVG) format, as well as simple editing functions. This stylesheet concept has a lot of potential for handling customization, which we discuss in the later parts of the paper.

7.1.3 *Ontorama*

Ontorama is a visualization tool for RDF-based graphs, detailed in Eklund *et al.* Eklund *et al.* (2002). It presents RDF graphs in a hyperbolic graph layout using radial layouts to align the nodes on a spherical surface. A significant challenge for Ontorama and other hyperbolic browsers is that not all ontologies are trees (in the mathematical sense) according to the inheritance hierarchy (is-a). For example, some domain models are constructed using parthood (part-of) as the key structural relationship. This means these tools must somehow handle the case where these relationships are discontinuous in order to display all the nodes—i.e., be able to visualize forests as well as trees.

7.1.4 *Ontobroker/Kaon*

The Ontobroker tool (Decker *et al.*, 1999) uses a similar hyperbolic view technique to aid in the navigation of ontologies. It has recently been superseded to some extent by KAON (Maedche *et al.*, 2003), a similar tool with more of a focus on the Semantic Web. The strengths of these tools lies in the degree of integration between the tool and the visualization engine, which makes the representations in the graphs more salient.

7.2 *How existing visualizations in Protégé address cognitive support needs*

We now make a more detailed analysis of the existing visualizations provided in Protégé at the time of writing. One additional tool, the Ez-Owl tab available at <http://iweb.etri.re.kr/ezowl/index.html>, deals mainly with Description Logic ontologies and is not discussed here.

We summarize the essential features of each tool, followed by identifying where each tool meets or fails to meet the areas identified in Section 6. Table 2 summarizes these findings. To show examples, we use a well-known ontology describing the domain of wines, first used in a paper describing the CLASSIC knowledge representation system (Brachman *et al.*, 1991), and now also in other examples, such as the Web Ontology Language guide (see <http://www.w3.org/TR/owl-guide/>).

7.2.1 *Protégé core*

Protégé itself (shown in Fig. 1) has a fairly standard indented tree view interface, as shown in Fig. 1. Users can navigate the class tree by opening and closing class concepts, or by using a search panel located at the bottom of the screen. This default view meets several of the areas we identified, including top-down browsing

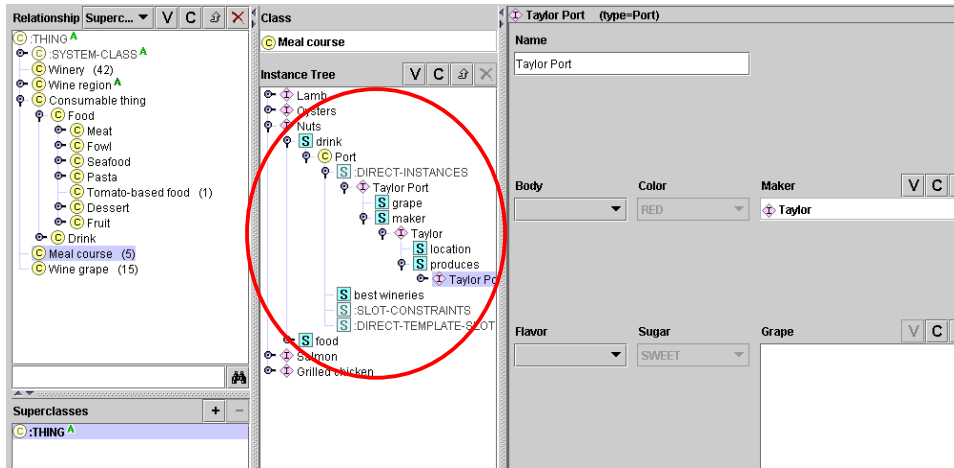


Fig. 2. Instance Tree tab for Protégé, supporting slot-based browsing

(task #1) and search visualization (task #3), to varying degrees. The chief limitation of the default Protégé interface makes it very difficult to grasp more complex relationships, such as multiple inheritance or complex slot compositions (as used in reification). Particularly interesting is the lack of support in core Protégé for the verification tasks. No methods are provided, for example, to understand complex relationships, or to see dependent concepts for a particular artifact of interest.

7.2.2 Instance Tree widget

This tool (shown in Fig. 2) extends the default Protégé interface. It allows users to browse through the classes using the slot values (related classes) defined in a particular concept, providing a means to perform slot-based browsing (task #2), as well as navigating between nodes during editing (task #6). Some support is also provided for navigating to arbitrary levels in the hierarchy. As a plug-in for Protégé, the widget leverages the other capabilities of Protégé to handle more complex tasks.

7.2.3 Ontoviz

Ontoviz, available at <http://protege.stanford.edu/plugins/ontoviz/ontoviz.html>, also relies on the GraphViz graph viewer to visualize ontologies in Protégé (see Fig. 3). Exploring the ontology is very difficult in Ontoviz, as the user interactions are restricted to panning and simple zooming (navigation tasks for top-down browsing (task #1)). Furthermore, OntoViz does not support more than one layout, which can only be based on the inheritance relationship. Customization is limited and the visualization does not scale beyond a few hundred entities. Ontoviz, as shown by the task support it provides, lacks a lot of support typical of more interactive tools; for example, one cannot browse multiple relationships.

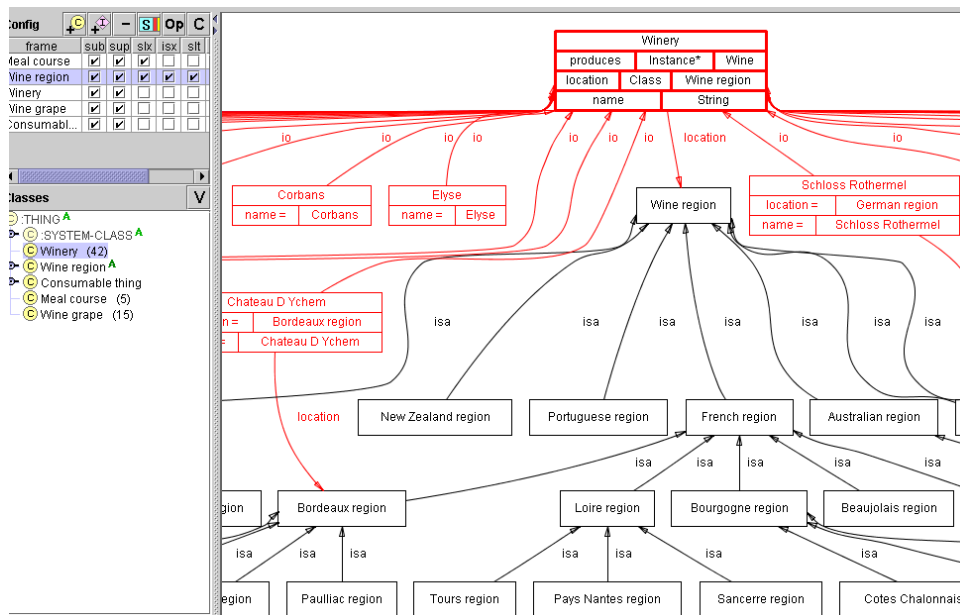


Fig. 3. Ontoviz plug-in for Protégé, showing a portion of the wines ontology

7.2.4 TGVizTab

Developed in 2002 by Harith Alani (Alani, 2003), TGVizTab (Fig. 4) makes use of a spring-embedding algorithm to implement a customizable layout for concepts and relationships. The true strength of Touchgraph lies in its salience. The graph shown is typically readily apparent to users. Some of the weaknesses of TGVizTab concern the difficulty of seeing all relationships (e.g., non-structural relationships), screen clutter, and the difficulty in synchronizing this view with the Protégé view. TGVizTab does support showing search results, and is excellent for incremental browsing of the ontology. The user can easily turn on or off certain nodes and arcs for an exploration of interest. It is not suitable for top-down exploration, because the graph is fairly unordered. One can save views, but only in the Touchgraph (XML) format.

7.2.5 Jambalaya

Jambalaya, developed by our research group and shown in Fig. 5, is a suite of tools and views for viewing ontologies with graph metaphors; there are several different mechanisms for viewing data in Jambalaya. As mentioned, Jambalaya uses a graphical language based on the mathematical theory of graphs to map from the Protégé meta-model (based on frame-based knowledge representations) to a visual representation. Jambalaya excels at supporting navigation tasks, although browsing search results is not always simple. Like the other tools, Jambalaya has little or no support for editing tasks. In the verification tasks, Jambalaya does fairly well; one can abstract information and search for arbitrary levels of interest, and identifying incoming relationships is easy as well. While there is no support for

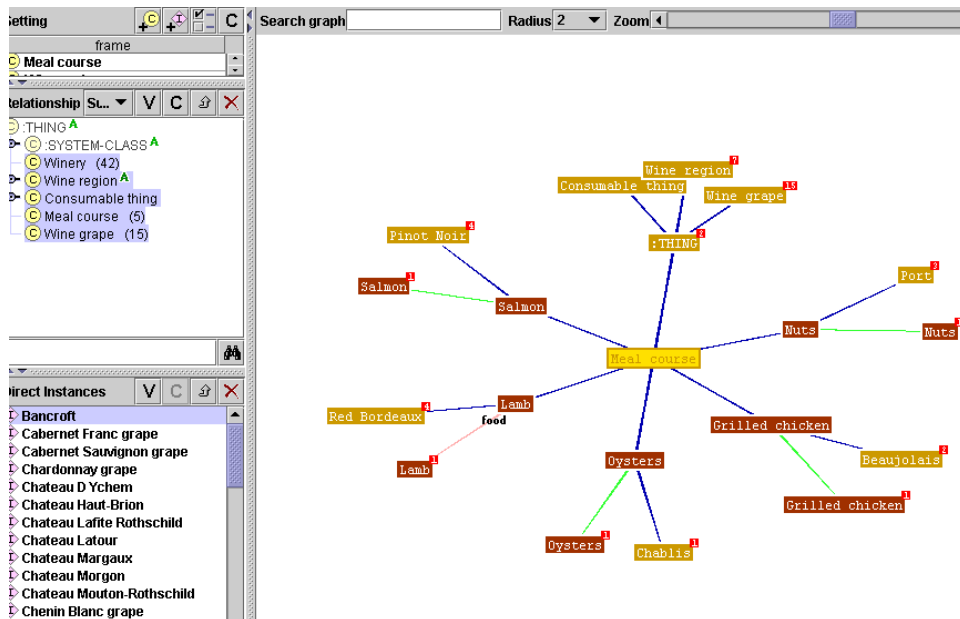


Fig. 4. TGvizTab plug-in for Protégé, using a hyperbolic layout on the wines ontology

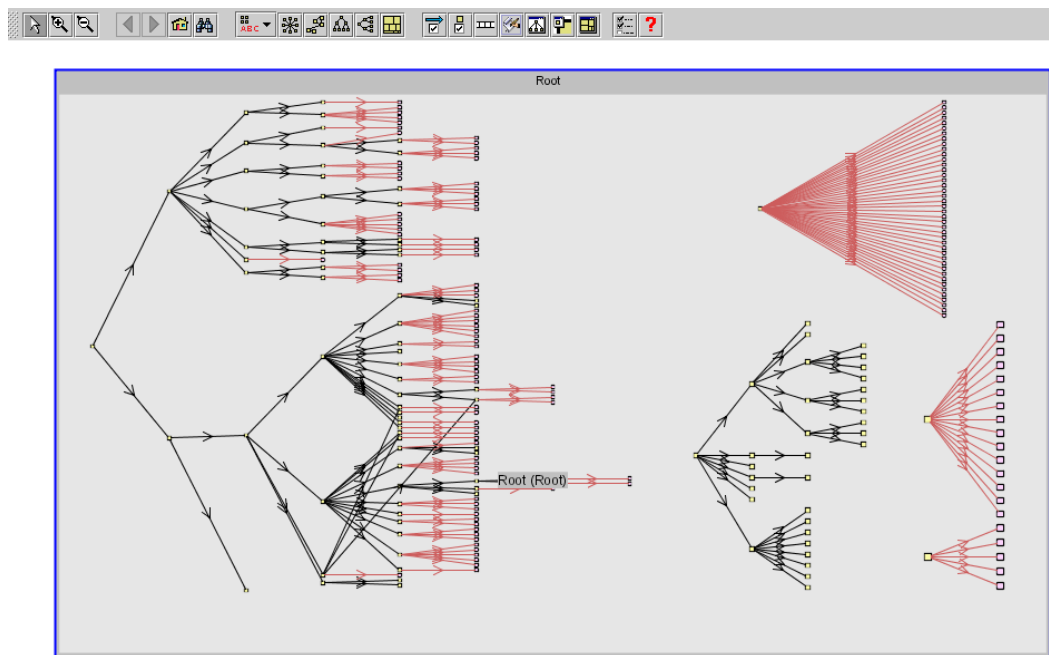


Fig. 5. Jambalaya plug-in for Protégé, showing the concepts and relations in the wines ontology

incremental navigation, there is rudimentary support for complex relationships: one can see which hierarchies a node belongs to, but there is no support for reification.

7.3 Summary

Based on the areas where cognitive support is required (as listed in Section 6), we can conclude certain things about the five tools we examined. The tools do not support editing tasks, particularly graphical modeling; this probably reflects the focus on ontology comprehension, and not editing, that these tools have. While all tools provide different but complementary approaches to navigating and verifying domain models, the lack of certain cognitive support hinders the utility for real-world users. Table 2 outlines the cognitive support offered by the five tools.

Category	Area of Cognitive Support	Protégé	Inst. Tree	Ontoviz	TGViz	Jamb.
Navigation	1. Overviews	x	p	x	p	x
	2. Slot-based browsing	p	p	-	x	x
	3. Show queries	p	-	-	x	p
	4. Save views	-	-	x	p	x
Editing	5. Graphical editing	x	-	-	-	-
	6. Editing navigation	-	x	-	-	-
	7. Ontology reuse	p	-	-	-	-
Verification	8. Incoming relations	-	-	x	x	x
	9. Incremental navigation	-	x	-	x	-
	10. Multiple relations	-	-	-	-	p

Table 2

List of Protégé and its extensions evaluated against knowledge engineering tasks. An **x** indicates the area was determined from that research method, a **p** that there was partial support, and a dash that there was no support.

8 Evaluating cognitive support using non-functional design goals

We have described some tasks in knowledge engineering which require cognitive support and outlined them in Table 1. We then examined the cognitive support

that was provided by a popular knowledge engineering tool, Protégé, and some of its related extensions (as shown in Table 2). This section presents some thoughts on how designers can evaluate the cognitive support in a particular tool (and for a particular domain). Non-functional requirements (Mylopoulos *et al.* (1992), Boehm (1988)), also known as quality attributes, constraints, and effectiveness criteria, are high-level objectives such as scalability, usability, and customizability, which can be thought of as goals which guide the design of features in a system, particularly in terms of what support is provided for the functional requirements or tasks (Chung *et al.*, 1995). These goals are often conflicting, in much the same way as design goals for knowledge representation languages are. For example, just as language designers must balance the goals of expressiveness and tractability, designers of cognitive support tools for knowledge engineering must balance scalability and usability. As such, design of these tools can be thought of as a series of trade-offs and architectural decisions which attempt to balance these goals. The following paragraphs use Jambalaya as a case study to analyze these decisions.

8.1 Trade-offs in the design process

As we mentioned in the introduction to this section, the design of cognitive support tools for knowledge engineering involves determining which non-functional goals or constraints the system should address, in addition to decisions on what tasks to support. Choosing which goals to focus development efforts on involves a series of choices about which goals are more important to the tool as well as any domain-specific needs. The level of focus determines to what extent each goal is met, or at least, to what extent the developers care about it (as it may not be met). Goals exist on a continuum, with a range of concerns possible.

Some examples of such trade-offs are listed in Ahlers and Weimer (2002), who detail certain trade-offs that visualizations of knowledge-based systems have to contend with. These trade-offs cover both functional task requirements and non-functional design goals of the tool. For instance, a tool should provide *human readable* and comprehensible information and views, while still allowing for semantically rich, *machine-processable* data. Both can be considered functional tasks the tool needs to address. Showing sufficient *local detail* while still giving the user adequate *global context* requires designers understand what focus their tool will have for these functional tasks – customizability is an important consideration for this trade-off. In a similar vein, a tool should give an *overview* to abstract the underlying *complexity*, but sometimes that complexity is necessary and essential. Sometimes what is important is the *structure* of the model; at other times, the *content* is more important. As mentioned in Section 8.5, users typically want systems which can handle large problems – *scalability* – but can still maintain adequate response times (*usability*), trade-offs in design goals. A system should also maintain its *flexibility* to support the user's unique mental approach to his task, but it should strive

to maintain *consistency* as well - for example, always showing key elements at the same locations. This represents a trade-off in non-functional design constraints and must be decided based on the domain the tool will address. Finally, another non-functional trade-off is that in some cases, *known metaphors* are easily understood and most usable, but at times only *innovative approaches* can address the task.

This small example of some design trade-offs illustrates the complexity of designing a tool which provides cognitive support for knowledge engineering tasks. Not only should those tasks be identified and supported, but developers must keep in mind the non-functional goals of the tool, such as how well it scales, whether it can be adapted, and how usable its interface is. Clearly, this set of factors produces a large number of solutions which can be created. Based on the findings of this paper, we believe that in practice a well-designed tool, with a carefully determined set of functionality, will be able to accommodate many disparate domains and users.

An earlier work, by Eisenstadt *et al.* (1990), also presents design goals of knowledge engineering tools based on their work in program visualization and visual programming, and for completeness, we also refer to that work in the following analysis. We discuss five such goals and how they impact the provision of cognitive support in knowledge engineering tools. The major elements we have identified include:

- Usability
- Learnability
- Expressivity
- Scalability
- Customizability and extensibility

For brevity, we refer only to how Jambalaya meets these goals in this section, although they apply to all the tools we have looked at. There is no definitive list of non-functional requirements, as this is antithetical to what they represent – domain specific goals that are often only identified as the system is implemented (suggesting an iterative, spiral model (Boehm (1988)) of development might be best).

8.2 Usability

Another important criteria is that of usability, which supports users staying ‘in the flow’ (Bederson, 2003), that is, staying on-task and not being unnecessarily distracted with interface issues. Usability affects tools by constraining the complexity and variety of metaphors and techniques a tool can use to provide the cognitive support; there is often a direct relationship between scalability and usability. Usability is best assessed by conducting user evaluations. Jambalaya has made some strides towards improved usability, including some preliminary user studies, a heuristic evaluation of the interface, and informal evaluations, but traditionally, this was not

a big focus. This is now changing as we realize that neglecting usability requirements often impacts tool use – preventing users from accessing the task-specific cognitive support we aim to provide. Eisenstadt *et al.* also mention some usability principles for knowledge engineering tools, including ensuring there is a mapping between what is shown and what is present in the model, and that all views should be coupled so that changes in one are shown in the others. They also mention that tools should be able to abstract and switch views, and objects in a view should be manipulable.

8.3 Learnability

This design goal focuses on making the key tools and functionality in the application readily apparent. Whereas usability is focused on making the tool itself easy to use, learnability requires that the functions that will most help a user be salient and visible (also mentioned in Eisenstadt *et al.* (1990)). For example, in the NCI domain, a useful visualization in Jambalaya is to perform a spring layout on the `has-biological-process` relation, which shows which genes or proteins play roles in biological processes. Jambalaya does not make this powerful visualization apparent, since its design was focused more on the expressive power of the tool. Learnability is also related to the customizability of a tool, since, as demonstrated, different domains and different users will have various sets of functionality which would be most useful.

8.4 Expressivity

Expressivity has typically been used in the sense of formal languages, such as first-order logic, to denote what the language can represent. In tools with user interface components, such as the ones discussed here, expressivity refers to what aspects of a model can be represented in the tool (refer to Clark *et al.* (2001)) for another example). Eisenstadt *et al.* call this completeness – a tool should ideally show all implications of a model (and only those implications). Expressivity often has an inverse relationship with scalability. For example, showing all statements that are implied by a description logic language may not even be computable, so tools working with, for example, OWL ontologies, need to be aware of these constraints. In Protégé, there is also a notion of system classes, which are meta-objects used to construct a domain of discourse. Initially Jambalaya did not represent these objects, yet they can have major implications for modelers: the FMA project, for example, uses metaclasses extensively. Jambalaya now addresses the non-functional constraint of expressivity by allowing users to determine whether they wish to see these classes. However, it is still not clear to us how to support the user in knowing which concepts to render in a visualization.

8.5 Scalability and responsiveness

A very important criteria in many tools in the knowledge engineering domain is scalability, the ability of a tool to handle large problems. Cognitive support is ineffective if it cannot deal with the problem of interest, which may be quite large; the NCI ontology consists of approximately 40,000 concepts. We are working on improving the responsiveness of Jambalaya so that using it with reasonable sized ontologies (currently around one hundred thousand artifacts) does not affect one's work patterns. We are also examining how different visualization metaphors and toolkits can be used to rapidly visualize changes between versions of ontologies (to better support navigation and verification tasks). Jambalaya provides a range of views that the user can select from, rather than one or two metaphors. The reason we focus less on innovative views on the data is that we believe that the techniques are not the major hurdle in people's use of the tool; often, the issue is related to one of the design goals we have discussed, such as scalability or usability. Scale can have a negative impact on how a metaphor is used, and much of the design challenge lies not in determining what useful metaphors might be, but rather, in how to adapt metaphors for different problems (for example, by using abstraction techniques).

8.6 Customizability and Extensibility

Implicit in our belief that there is great variability in user needs, often due to changing domains, is the fact that users are the ones best able to decide what they need. The *customizability* criteria measures how well a tool allows an end-user (in this domain, a modeler) to tailor a tool's functional support to his or her needs. It is very difficult for designers to determine what cognitive support for which tasks is necessary in all cases. Jambalaya partly addresses the customizability goal with rudimentary scripting support to provide access to the application domain. We have also created a user-modifiable ontology that describes the Jambalaya tool domain (the set of visualizations, tools, and elements in Jambalaya). Modifications will allow the application to be customized by either end-users (modelers at the NCI, for example) or their technically proficient colleagues (who could be termed Gurus (Michaud and Storey, 2003)). This customization may take the form of hiding additional complexity that is not needed, by removing unwanted tools or views, or by writing custom scripts. We are also interested in making Jambalaya itself more flexible at the source code level by adding some form of extension points, similar to the extension features Protégé currently offers. This could allow, for instance, other teams to develop plug-ins for Jambalaya that offer new views or layouts. We would like to take some ideas from the CODE4 tool mentioned in the background section, and make the distinction between the model elements of Jambalaya and the views on those elements even more pronounced.

Focusing on customization has one main advantage. Despite some commonalities, such as the need for slot-based browsing, editing support, and incremental navigation, individual domains require subtle yet important changes to standard functionality. Visual interfaces are difficult and time-consuming to construct. In order to have a realistic chance of being adopted, this design must be done on a case by case basis. For example, the visualizations needed by the NCI team will be different than those needed by the Foundational Model of Anatomy team. The FMA modelers need a tool which can show complex relationships between meta-model and model, a tool which shows modeling errors of commission or omission. While requiring similar task support, the NCI modelers are more interested in the location of new concepts in the hierarchies. This distinction involves differences in the degree to which a tool supports specific tasks.

The notion of domain-dependent tasks may seem to contradict our earlier suggestions that there are certain tasks common to knowledge engineering which all tools should support. However, this is not so. While there are tasks common to all projects, such as top-down navigation or slot-based browsing, it is in the details of *how* these tasks are supported, and, as mentioned in the preceding chapter, to what degree design goals are examined, that determine what a tool will look like, and to what degree it will be adopted.

Considering customization as a high-priority design goal allows tools to address these different degrees of interest, without having to make these decisions at design time. For example, incorporating mechanisms to support feature selection and enhancement allows the tool to be tailored to individual domains. In the NCI's case, they have suggested this take the form of a customized version of Jambalaya, along with useful end-user scripts. These tools would be posted on a web site for interested users of their large ontology as a 'suggested' way to interact with the ontology.

We have recently implemented a prototype version of customization in Jambalaya designed to evaluate this. This extension models the elements of Jambalaya in an ontology – such as visualization tools and editing functions – and allows domain customizations using Protégé's standard ontology editing features. The domain experts can select Jambalaya features, such as horizontal tree layouts and the relationships first shown upon start-up, by editing an included Protégé ontology. Our qualitative experience reports from five expert knowledge modelers indicate that the customization approach, while powerful, relies too heavily on domain experts to understand the visualization tool. This work is discussed in more detail in Ernst (2003).

9 Conclusion

We have discussed the need for improved cognitive support in knowledge engineering tools, with specific reference to Protégé; we believe that this support will be even more necessary as ontologies invariably become larger. We detailed areas of cognitive support for common knowledge engineering tasks, basing them on our own background work, several existing studies, and the qualitative research we have conducted at two large-scale projects. While they may be preliminary, we firmly believe that the tasks represent a good initial description of the problems faced and needs users have for these tools. Although we concentrated on identifying requirements for modelers, we believe the requirements are also applicable to other types of users.

We then evaluated several tools according to this framework; although the tools were derived from the field of information visualization, we believe information visualization is essentially an advanced user interface, and as such, these criteria can be applied to other tools. There are other interfaces, such as those for database systems, that could be leveraged in this domain. Finally, we evaluated the functional support one of the tools (Jambalaya) provided by evaluating it in the context of non-functional design goals. This evaluation also serves as an example of how this approach might be applied to other tool. It emphasizes the need for tool designers to have a comprehensive understanding of what cognitive support a tool is trying to provide, and to whom it is being provided.

We wish to conclude by issuing a call for action: the challenges we have detailed, in conjunction with the current support lacking in many tools, make the issue of cognitive support for knowledge engineering tools a crucial one.

10 Acknowledgments

We wish to acknowledge the many people who have made this work possible, notably the members of the CHISEL research group. Thanks also to the external reviewers for their invaluable comments. Funding for this research was provided by an NSERC scholarship, and a grant from the National Cancer Institute. We wish also to acknowledge the support, financial, moral, and otherwise, from the Protégé team at Stanford, including Dr. Mark Musen, Dr. Natalya Noy and Dr. Ray Ferguson. The Protégé resource is supported, in part, by grant P41 LM007885 from the National Library of Medicine.

References

- Ahlers, J., Weimer, H., 2002. Challenges in interactive visualization for knowledge management. In: Sixth International Conference on Information Visualisation (IV02). IEEE, London, UK, pp. 367–371.
- Alani, H., 2003. TGVizTab: An ontology visualisation extension for Protégé. In: Knowledge Capture 03 - Workshop on Visualizing Information in Knowledge Engineering. ACM, Sanibel Island, FL, pp. 2–7.
- Allen, M. M., 2003. Empirical Evaluation of a Visualization Tool for Knowledge Engineering. M. Sc. thesis, University of Victoria.
- Bederson, B., October 2003. Interfaces for staying in the flow. Technical Report HCIL-2003-37, University of Maryland Human Computer Interaction Lab.
- Berners-Lee, T., Hendler, J., Lassila, O., May 17 2001. The semantic web. *Scientific American*, 279.
- Beyer, H., Holtzblatt, K., 1998. Contextual Design: Defining Customer-Centred Systems. Morgan Kaufmann, San Francisco.
- Blythe, J., Kim, J., Ramachandran, S., Gil, Y., 2001. An integrated environment for knowledge acquisition. In: Int. Conf. on Intelligent User Interfaces. San Francisco, CA, pp. 13–20.
- Boehm, B., 1988. A spiral model of software development and enhancement. *IEEE Computer* 21 (5), 61–72.
- Brachman, R., McGuinness, D. L., Patel-Schneider, P., Resnick, L., Borgida, A., 1991. Living with classic: When and how to use kl-one-like language. In: Sowa, J. F. (Ed.), *Principles of Semantic Networks*. Morgan Kaufmann, pp. 401–456.
- Card, S., Mackinlay, J. D., Shneiderman, B., 1999. *Readings in Information Visualization: Using Vision to Think*. Academic Press, London.
- Chung, L., Nixon, B., Yu, E., 1995. Using non-functional requirements to systematically select among alternatives in architectural design. In: 1st International Workshop on Architectures for Software Systems. Seattle, pp. 31–43.
- Clark, P., Thompson, J., Barker, K., Porter, B., Chaudhri, V., Rodriguez, A., Thomere, J., Mishra, S., Gil, Y., Hayes, P., Reichherzer, T., 2001. Knowledge entry as the graphical assembly of components. In: 1st International Conference on Knowledge Capture (K-Cap '01). ACM Press, Victoria, BC, pp. 22–29.
- Clitherow, P., Riecken, D., Muller, M., 1989. Visar: a system for inference and navigation of hypertext. In: Conference on Hypertext and Hypermedia. ACM, Pittsburgh, Pennsylvania, United States, pp. 293 – 304.
- Decker, S., Erdmann, M., Fensel, D., Studer, R., 1999. Ontobroker: Ontology based access to distributed and semi-structured information. In: Meersman, R. (Ed.), *DS-8. Semantic Issues in Multimedia Systems*. Kluwer, pp. 351–369.
- Devedzic, V., 2002. Understanding ontological engineering. *Communications of the ACM* 45 (4), 136–144.
- Eisenstadt, M., Domingue, J., Rajan, T., Motta, E., 1990. Visual knowledge engineering. *IEEE Trans. on Software Engineering* 16 (10), 1164–1177.
- Eklund, P., Roberts, N., Green, S. P., 2002. Ontorama: Browsing an rdf ontology using a hyperbolic-like browser. In: First International Symposium on Cyber-

- Worlds (CW2002). IEEE, Tokyo, pp. 405–411.
- Ernst, N. A., 2003. Towards cognitive support in knowledge engineering: An adoption-centred customization framework for visual interfaces. M. Sc. thesis, University of Victoria.
- Ernst, N. A., Storey, M.-A., August 19 2003. A preliminary analysis of visualization requirements in knowledge engineering tools. Chisel technical report, University of Victoria.
- Fairchild, K., Poltrock, S., Furnas, G., 1988. Semnet: Three-dimensional graphic representations of large knowledge bases. In: Guidon, R. (Ed.), *Cognitive Science and its Applications for Human-Computer Interaction*. Lawrence Erlbaum Associates, pp. 201–233.
- Falbo, R. d. A., Guizzardi, G., Duarte, K. C., 2002. An ontological approach to domain engineering. In: *International Conference on Software Engineering and Knowledge Engineering, SEKE02*. Ischia, Italy.
- Fluit, C., Sabou, M., Harmelen, F. v., 2002. Ontology-based information visualisation. In: Geroimenko, V., Chen, C. (Eds.), *Visualising the Semantic Web*. Springer-Verlag.
- Gaines, B., Shaw, M., 1995. Concept maps as hypermedia components. *Int. Journal of Human-Computer Studies: Special Issue on Knowledge-Based Hypermedia* 43 (3), 323–361.
- Gennari, J. H., Musen, M. A., Ferguson, R., Grosso, W. E., Crubzy, M., Eriksson, H., Noy, N. F., Tu, S. W., 2003. The evolution of Protégé: An environment for knowledge-based systems development. *International Journal of Human-Computer Studies* 58 (1), 89–123.
- Ginsberg, M., 1991. Knowledge interchange format: The KIF of death. *AI Magazine* 12 (3).
- Golbeck, J., Frago, G., Hartel, F., Hendler, J., Parsia, B., Oberthaler, J., 2003. The National Cancer Institute's thesaurus and ontology. *Journal of Web Semantics* 1 (1).
- Hendler, J. (Ed.), 1988. *Expert Systems: The User Interface*. Human/Computer Interaction. Ablex Publishing, Norwood, NJ.
- Kalfoglou, Y., 2000. Exploring ontologies. In: Chen, C. (Ed.), *Handbook of Software Engineering and Knowledge Engineering*. Vol. 1. World Scientific Publishing Company.
- Karp, P. D., Chaudhri, V. K., Paley, S. M., 1999. A collaborative environment for authoring large knowledge bases. *Journal of Intelligent Information Systems* 13, 155–194.
- Knublauch, H., Musen, M. A., Noy, N. F., 2003. Creating semantic web (owl) ontologies with Protégé. In: *Int. Semantic Web Conference Tutorial*. Sanibel Island, Florida.
- Maedche, A., Motik, B., Stojanovic, L., Studer, R., Volz, R., March/April 2003. Ontologies for enterprise knowledge management. *IEEE Intelligent Systems* 18 (2), 26–33.
- Michaud, J., Storey, M.-A. D., 2003. The role of knowledge in software customization. In: *15th Int. Conf. on Software Engineering and Knowledge Engineering*

- (SEKE03). San Francisco Bay, CA.
- Mylopoulos, J., Chung, L., Nixon, B., 1992. Representing and using non-functional requirements: A process-oriented approach. *Software Engineering* 18 (6), 483–497.
- Ng, G. K.-C., 2000. Interactive visualization techniques for ontology development. Ph.d., University of Manchester.
- Richer, M., Clancey, W. J., 1985. Guidon-watch: A graphic interface for viewing a knowledge-based system. *IEEE Computer Graphics and Applications* 5 (11), 51–64.
- Shipman, F. M., Marshall, C. C., 1999. Formality considered harmful: Experiences, emerging themes, and directions on the use of formal representations in interactive systems. *Computer Supported Cooperative Work* 8 (4), 333–352.
- Skuce, D., Lethbridge, T. C., 1995. CODE4: A unified system for managing conceptual knowledge. *International Journal of Human Computer Studies* 42, 413–451.
- Sowa, J. F., 2000. *Knowledge Representation: Logical, Philosophical, and Computational Foundations*. Brooks/Cole, Pacific Grove, CA.
- Stelzner, M., Williams, M. D., 1988. The evolution of interface requirements for expert systems. In: Hendler, J. (Ed.), *Expert Systems: The User Interface*. Human/Computer Interaction. Ablex Publishing, Norwood, NJ, pp. 285–306.
- Storey, M.-A., Fracchia, F., Müller, H. A., 1999. Cognitive design elements to support the construction of a mental model during software exploration. *Journal of Software Systems: special issue on Program Comprehension* 44, 171–185.
- Storey, M.-A. D., Musen, M. A., Silva, J., Best, C., Ernst, N., Ferguson, R., Noy, N. F., 2001. Jambalaya: Interactive visualization to enhance ontology authoring and knowledge acquisition in Protégé. In: *Workshop on Interactive Tools for Knowledge Capture, K-CAP-2001*. Victoria, B.C. Canada.
- Storey, M.-A. D., Wong, K., Fracchia, F., Müller, H., 1997. On integrating visualization techniques for effective software exploration. In: *InfoVis '97*. Phoenix, AZ, pp. 38–45.
- Studer, R., Benjamins, V. R., Fensel, D., 1998. Knowledge engineering: Principles and methods. *Data Knowledge Engineering* 25 (1-2), 161–197.
- Tallis, M., Kim, J., Gil, Y., 1999. User studies of knowledge acquisition tools: Methodology and lessons learned. In: *Knowledge Acquisition Workshop*.
- Travers, M., 1989. A visual representation for knowledge structures. In: *2nd ACM Conf. on Hypertext*. ACM, Pittsburgh PA, pp. 147–158.
- Walenstein, A., 2002a. Cognitive support in software engineering tools: A distributed cognition framework. Ph.d. dissertation, Simon Fraser University.
- Walenstein, A., 2002b. Foundations of cognitive support: Toward abstract patterns of usefulness. In: *14th Annual Conference on Design, Specification, and Verification of Interactive Systems (DSV-IS'2002)*. Rostock, Germany.