

## Chapter 2

# ENGINEERING AND CUSTOMIZING ONTOLOGIES

### *The Human-Computer Challenge in Ontology Engineering*

Martin Dzbor and Enrico Motta

*Knowledge Media Institute, The Open University, UK, {M.Dzbor, E.Motta}@open.ac.uk,  
Tel. +44-1908-653-800; Fax +44-1908-653-169*

**Abstract:** In this chapter we introduce and then discuss the broad and rather complex area of human-ontology interaction. After reviewing generic tenets of HCI and their relevance to ontology management, we give an empirical evidence of some HCI challenges for ontology engineering tools and the shortcomings in some existing tools from this viewpoint. We highlight several functional opportunities that seem to be missing in the existing tools, and then look at three areas that may help rectifying the identified gaps. We relate methods from user profiling, large data set navigation and ontology customization into a “triple stack,” which may bring tools for engineering ontologies from the level of niche products targeting highly trained specialists to the ‘mainstream’ level suitable for practitioners and ordinary users. The work presented in this chapter is based on the authors’ research together with other colleagues in the context of the “NeOn: Lifecycle Support for Networked Ontologies” project.

**Keywords:** HCI; human-ontology interaction; NeOn; networked ontologies; ontology customization; user study of ontology engineering tools

## 1. INTRODUCTION

Human-computer interaction (HCI) is a well-established and rich subject that has an impact not only on those who develop computational systems, but also on the users of such systems, the vendors, maintainers, and many more stakeholders who are normally involved in designing and delivering software and computer-based tools. At the centre of HCI as a science is the core of its investigation: *interactions*. Note that this emphasis on an abstract notion “interaction” does not reduce the importance of the users or push them into a background.

On the contrary, the term “interaction” is broader, and in general, involves three constituting parts: *the user*, *the technology*, and *the way they work together*. One can then study such phenomena as how the users work with a particular technology, what the users prefer, how the technology addresses given issues, etc. The purpose of this chapter is not to delve into generic HCI issues applicable to any technology. We want to expand the views of HCI to cover what we label as human-ontology interaction.

Human-ontology interaction can be seen as a subset of HCI issues that apply to specific tasks and specific technologies. Our aim is to investigate how users interact with the ontologies, in general, and with *networked* ontologies, in particular, and how they do it in a realistic ontology lifecycle scenario. While HCI is a subject almost as old as the computer science, the specifics of interacting with ontologies were not considered in much depth. Tools supporting ontological engineering are considered to be primarily software tools, and thus, it is presumed that general findings of the HCI practitioners also apply to ontologies.

To some extent, this is true; however, design, engineering and subsequently maintenance of ontologies are indeed specific ways to interact with the technology. In other words, the change in the activity implies a change in the entire interaction. Thus, an action that may look similarly to other software systems (e.g. opening a file) may acquire semantically very specific meaning in the context of a particular activity (in our case, ontology engineering).

In this chapter, we look at several different aspects of how a user may interact with ontologies in a varied sort of ways. The first part of the chapter is concerned with a user study that we carried out in order to improve our understanding of the level of user support provided by current ontology engineering tools in the context envisaged by the NeOn project<sup>1</sup>. That is, in a scenario when ontology engineers are developing complex ontologies by reuse, i.e., by integrating existing semantic resources.

While the existing empirical work on exploring HCI aspects of the ontology engineering tools points to several problems and challenges, we decided to conduct a new study, because none of the studies reviewed in section 2.1 provided sufficient data to drive the development of the ontology engineering tools addressing the NeOn scenario. In particular, the use of tools by ordinary users, the emphasis on ontology reuse and the embedment of the study in a real-world engineering task.

A complementary view to this empirical user study is presented in the latter part of the chapter: exploring the HCI challenge with more analytic

---

<sup>1</sup> “NeOn: Lifecycle support for networked ontologies” is a large-scale integrated project co-funded by the European Commission by grant no. IST-2005-027595; more information on its focus, outcomes and achievements so far can be found on <http://NeOn-project.org>.

lenses, and focusing on a variety of tools that were specifically designed to support ontological engineering, or could be reused with ontologies in a serendipitous manner. With this view in mind we consider several approaches, technologies, and tools to illustrate various aspects of where user interaction with ontologies becomes somewhat specific and different from using other software systems and tools.

Before going more in depth, let us introduce the basic terminology first. In order to work in a structured manner, we separate the terms that traditionally come from the HCI domain from the terms that are typical for ontology engineering.

## 1.1 Terms frequently used in HCI

In this section we present common and established meanings of terms and issues that are usually mentioned in connection with user interaction in general. The purpose of this brief glossary is twofold: (i) to introduce terms that are used in the subsequent sections of this chapter to those practitioners with less background in traditional HCI, and (ii) to differentiate between terms that are often used interchangeably by lay persons. We are not defining here any terms related to ontology engineering in general, as these have a broader scope of validity than the chapter on HCI challenges, and are covered elsewhere in the book.

- **Accessibility:** In general, this term reflects the degree to which a given system is usable by different users. It can be expressed in terms of ease with which to access certain features or functions of the system, together with the possible benefits such access may bring to the user. Often this term is interpreted in the sense of ‘enabling people who are physically disabled to interact with the system.’ This is a slightly unfortunate emphasis on one specific motivation for pursuing accessibility. In a non-disabled sense, accessibility may include aspects like appropriate language, jargon, level of detail, choice of action, etc.
- **Customization:** In the computer science this term refers to the capability of users to modify or otherwise alter the layout, appearance and/or content of information with which they want to interact. This term is often used together with personalization (see also explanation of term ‘profile’ below). In this deliverable we shall see customization as an ability to adapt user interfaces and tools so that they fit a particular user’s needs and accessibility constraints (see also term ‘accessibility’ above for some objective, explicit criteria that may be customized).
- **End user:** Popularly used to describe an abstract group of persons who ultimately operate or otherwise use a system — in computing, where this

term is most popular, the system corresponds to a piece of software. The abstraction is expressed in terms of a relevant sub-set of a user's characteristics (e.g. his/her technical expertise, prior knowledge, task, objective, skill, etc.)—leading to such user categories as knowledge engineers, developers, administrators, etc.

- **Graphical User Interface (GUI):** GUI is a type of user interface that came to prominence in computer science in the 1980s. The hallmark of this type is the use of graphical images (so called widgets), texts and their managed appearance on the computer screen to represent the information and actions available to the user. Another hallmark is that the user's actions are performed by directly manipulating the graphical elements (widgets) on the screen. GUI is often defined in contrast with command-based, text-only or terminal-based user interfaces.
- **Localization:** In the context of computing and HCI, localization is seen as the adaptation of an object or a system to a particular locality. A typical example is where a locality is defined in terms of different languages (e.g. English, Spanish, etc.), and the system is expected to translate messages and other aspects of its UI into the language suitable for or selected by the user. Thus, localization may be seen as a customization of a tool for a specific country, region or language group. In some literature, this term is used jointly with term 'internationalization.' However, language is only one (albeit most visible) aspect of the system UI that can be translated to the local customs. Other aspects that may need amendments include issues like time and date formatting, decimal number formatting, phone and postcode formatting, and locally used units of measure (e.g. feet, meters, etc.) Less common adaptations are in the use of colors, layouts and imaging appropriate to a particular locality.
- **Modality (of user interface):** A path or communication channel employed by the user interface to accomplish required inputs, outputs and other activities. Common modalities include e.g. keyboard, mouse, monitor, etc.
- **(User) Preference:** This term represents a real or imagined choice between alternatives and a capability to rank the alternatives according to some criterion. In computer science, this term is typically used in the sense that users choose among alternative user interactions, user interface components and/or paths. In computing, user preferences are often based on the utility (value) of the available alternatives to the particular user, in a particular situation or task.
- **(User) Profile:** a term seen in the context of computing as a way to describe some user properties that are relevant for a particular task and can help in tailoring information delivery to the specific user. Note that

‘user’ may mean a concrete individual person as well as an abstract user (e.g. a group or type).

- **Usability:** A degree to which the design of a particular user interface takes into account human needs defined in terms of psychology or physiology of the users. Usability looks at how effective, efficient and satisfying the user interface (and the underlying application) is.
- **User experience:** Broadly, this term describes an overall experience, satisfaction and/or attitude a user has when using a particular system. In computing, this term is often used interchangeably with terms like usability and sometimes accessibility.

## 1.2 About ontological engineering

In the early 1990’s, a group of Artificial Intelligence (AI) and database (DB) researchers got together to define a standard architecture stack for allowing intelligent systems to interoperate over a knowledge channel and share data, models, and other knowledge without sharing data schema or formats. This group comprised Tom Gruber—the person who is widely credited with clarifying a definition of ontology for the AI community and for promoting the vision of ontologies as enabling technology:

*“In the context of knowledge sharing, I use the term ontology to mean a specification of a conceptualization. That is, an ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. This definition is consistent with the usage of ontology as set-of-concept-definitions, but more general.”* (Gruber 1993a; Gruber 1993b)

Ontologies are designed artifacts, similar to cars, desks or computers. As such, they always have a purpose, they are *engineered for something*. In the original vision of Tom Gruber, ontologies were artifacts facilitating sharing and interchange of knowledge, or making commitments to particular meanings. While an ontology may be in principle an abstract conceptual structure, from the practical perspective, it makes sense to express it in some selected *formal language* to realize the intended shareable meaning.

Such formal languages then enable the negotiation of formal vocabularies, which, in turn, may be shared among parties in the knowledge sharing interaction without being dependent on either the user/agent or its context. One example of such a vocabulary may be description logic that allows us to make statements holding for some or all entities in a given world satisfying a given condition.

From the point of view of this book (and chapter), we often align the ontology management and engineering with the actual design, creation and

overall interaction with such formal vocabularies. If we take the Web Ontology Language (OWL<sup>2</sup>) as the current preferred formal vocabulary, then ontology engineering is often seen as a synonym to designing and coding conceptual commitment about the world or a particular problem in this language. Thus, for the purpose of this chapter, user challenge in engineering OWL ontologies is broadly definable as a user interaction with a particular software product, code, OWL model, OWL-based tool, technique, etc.

## 2. USERS IN ONTOLOGICAL ENGINEERING

In order to illustrate and ground the issues users are facing during the process of ontology design, engineering and management, this section includes extracts from a larger user study that has been conducted in the context of gathering and analyzing requirements in the NeOn project. The following sub-sections are based on our earlier workshop publication (Dzbor, Motta *et al.* 2006).

The existing empirical work on exploring HCI aspects of the ontology engineering tools highlights several problems with ontology engineering tools. However, at the beginning of the NeOn project we felt that there was a need to conduct a novel study, as none of the studies mentioned in section 2.1 provided the kind of data that can be used as a baseline to inform the development of the next generation ontology engineering tools.

### 2.1 Motivation and background

Some work on evaluating tools for ontology engineering has been done in the past. For example, Duineveld, Stoter *et al.* (2000) observed that the tools available in the time of their study (around 1999) were little more than research prototypes with significant problems in their user interfaces. These included too many options for visualizing ontologies, which tended to confuse the user and hinder navigation. Moreover, the systems' feedback was found to be poor, which meant a steep learning curve for non-expert users. Finally, most tools provided little support for raising the level of abstraction in the modelling process and expected the user to be proficient in low-level formalisms.

Pinto, Peralta *et al.* (2002) evaluated Protégé, one of the leading ontology engineering tools currently in use (Noy, Sintek *et al.* 2001), in several tasks, from the perspective of a power user. The authors found the system intuitive for expert knowledge engineers, as long as the operations were triggered by

---

<sup>2</sup> Specification of OWL as a W3C recommendation is on <http://w3.org/TR/owl-ref>

them (e.g. knowledge re-arrangement). However, difficulties arose when assistance from the tool was expected; e.g. in inference or consistency checks. Weak performance was also noted in language interoperability. In another survey, Fensel and Gómez-Pérez (2002) also noted issues with tool support for operations on ontologies beyond mere editing (e.g. integration or re-use). In particular, the authors emphasized the limited ‘intelligence’ of current tools—e.g. no possibility to re-use previously used processes in current design. Tools expected the user to drive the interaction, with the tool imposing constraints rather than adapting itself to users’ needs.

Yet another study by Storey, Lintern *et al.* (2004) focused on a fairly narrow aspect of visualization support in Protégé and its customization models are too complex and do not reflect users’ models of what they would normally want to see. Similar observations were made of the users having difficulties with description logic based formalisms in general (Kalyanpur, Parsia *et al.* 2005). Again, tools expected detailed knowledge of intricate language and logic details, and this often led to modelling errors.

As we mentioned earlier in the introduction, the existing empirical work on exploring HCI aspects of the ontology engineering tools highlighted several problems with ontology engineering tools. We conducted a new study, because none of the studies mentioned above provided the kind of data that can be used to inform the development of the ontology engineering tools envisaged by NeOn. Specifically, the studies did not satisfactorily address the following key concerns:

- **“Normal” users vs. “Power” users.** As ontologies become an established technology, it makes less sense to focus only on highly skilled knowledge engineers. There are so many organizations developing ontologies that it seems safe to assert that indeed most ontologies are currently built by people with no formal training in knowledge representation and ontology engineering. Therefore, it is essential to conduct studies, which focus on “normal users,” i.e., people with some knowledge of ontologies, but who are not classified as “power users.”
- **Emphasis on ontology reuse.** We adopt the view that ontologies will be networked, dynamically changing, shared by many applications and strongly dependent on the context in which they were developed or are used. In such scenario it would be prohibitively expensive to develop ontologies from scratch, and the re-use of existing, possibly imperfect, ontologies becomes the key engineering task. Thus, it makes sense to study the re-use task for OWL ontologies, rather than focusing only on a narrow activity (e.g. ontology visualization or consistency checking).
- **Evaluating formal ontology engineering tasks.** Studies reported earlier focused on generic tool functionalities, rather than specifically assessing

performance on concrete ontology engineering tasks. This creates two problems: (i) the results are tool-centric, i.e., it is difficult to go beyond a specific tool and draw generic lessons in terms of HCI on how people do ontology engineering tasks; (ii) by assessing the performance of our users on concrete tasks using OWL ontologies, we acquire robust, benchmark-like data, which (for example) can be used as a baseline to assess the support provided by other tools (including those planned in NeOn).

## 2.2 Overview of the observational user study

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

Ontology is, by definition, a shared artefact integrating views of different parties (Gruber 1993a). One form of integration used in this study was temporal, where an agent re-used previously agreed ontologies, perhaps from different domains. All studied ontologies were public; all were results of principled engineering processes and knowledge acquisition, and they all modelled domains comprehensible to a ‘normal user.’ The table shows some statistical information on the OWL ontologies included in the study.

*Table 2-1.* Descriptive features of the ontologies used in the evaluation study: numbers of primitives classified as **Cl**(asses), **Pr**(operties), and **Re**(strictions)

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

Two environments were used—Protégé from Stanford University<sup>3</sup> and TopBraid Composer from TopQuadrant<sup>4</sup>—these satisfied the initial

<sup>3</sup> Extensive details on the Protégé project and tool are available to an interested reader on <http://protege.stanford.edu>



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

### 2.2.1 Evaluation methodology

In our investigation of the ontology engineering environments, we opted for a formative evaluation (Scriven 1991). This choice was made mainly to inform design of new OWL engineering tools in the context of NeOn. Two constraints were observed: (i) gathered data shall not be tool-specific (it was not our objective to prove which one tool was best); and (ii) while generic tool usability was considered important, measures were expected not to be solely usability-centric. In terms of what was analyzed, we selected the following levels of analysis (Kirkpatrick 1994): (i) user’s satisfaction with a tool, (ii) effectiveness of a tool in achieving goals, and (iii) behavioural efficiency. In our study, these categories took the form of questions exploring usability, effectiveness, and efficiency categories, to which we added a generic functional assessment category.

Our questionnaire reflected situations that typically appear in the literature correlated with enhancing or reducing effectiveness, efficiency, usability or user satisfaction (Shneiderman and Plaisant 2004), and covered these situations by 36 questions. The remaining 17 questions inquired about various functional aspects considered relevant to the NeOn vision; including ontology re-use, visualization, contextualization, mapping, reasoning, etc.

The questionnaire included both open and closed (evaluative) questions. The former asked for opinions; the latter used a Likert scale ranging from very useful (+1) to very poor (−1). Each question was then expressed frequencies and counts—largely in the context of open, qualitative items and observations. Positively and negatively stated questionnaire items were interspersed to avoid the tendency of people to agree with statements rather than disagree (Colman 2001). Nevertheless, this tendency towards agreeing appeared during analysis; as was discussed in our preliminary report (Dzbor, Motta *et al.* 2006).

---

<sup>4</sup> More about TopBraid Composer can be found on <http://www.topbraidcomposer.com/>

### 2.2.2 User tasks

Participants were given three tasks considering different ways of integrating ontologies into a network. In Task 1, they were told that the *Copyright* ontology did not formalize temporal aspects, and had to be augmented with the relevant definitions from other ontologies (e.g. AKT Support). The objective was to review the three given ontologies, locate the relevant classes (i.e. *CreationProcess* and *Temporal-Thing*), import ontologies as needed, and assert that *CreationProcess* is a subclass of *Temporal-Thing*.

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

- **simple:** e.g. for concept *Economic-Rights* introduce statement *rangeOf( agent , Legal-Agent )*;
- **composite:** e.g. state that *rangeOf( recipient , ( Generic-Agent AND (¬ Geo-Political ) ) )*.

Task 3 asked people to re-define concept *copyright:Collective* so that formal statements could match an informal description. Participants were told to make amendments in the base — *Copyright* ontology, rather than to the other two. We expected they would first create new local sub-classes for the concept *copyright:Collective*, and then make them equivalent to the actual AKT classes. Task 3 also comprised a definition of a new property (e.g. *copyright:hasMember*) with appropriate domain and range, together with its restriction for class *copyright:Collective*, so that a collective is defined as containing min. 2 persons.

## 2.3 Findings from the user study

This section summarizes some findings from our study. For selected categories of measures we give a general summary of observations across the whole population, followed by commenting on differences (if any) between two common denominators of user performance in knowledge-intensive tasks — the choice of and the expertise with the tool. Particularly interesting is to look at how efficient people felt in different tasks, how they were assisted by the help system or tool tips, how the tools helped to navigate the ontologies or how easy it was to follow the formalisms used in

definitions. Table 2-2 shows general observations, and Table 2-3 compares features where differences between tools were observed.

The efficiency of the two tools was approximately the same. When asked about efficient handling of ontology dependencies and navigating through them, Protégé users thought they were significantly less efficient. Many users were not happy with the abstract syntax of the axiom formulae, which was not helped by the inability to edit more complex restrictions in the same windows and wizards as the simple ones.

*Table 2-2. Selection of a few general observations across population*

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

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

Measure/question	Type	Outcome	$\chi^2$	Sign
help with handling ontology dependencies	tools	TB (0.0) vs. Pr (-0.37)	7.65	yes
useful visualization & ontology navigation facilities	tools	TB (-0.33) vs. Pr (-0.63)	6.00	yes
handling ontology syntax / abstract syntax	tools	TB (+0.40) vs. Pr (-0.07)	2.33	no
ease/speed of carrying out integrations	experience	Le (-0.21) vs. Ex (+0.27)	9.75	yes
level of visualization and navigation support	experience	Le (-0.69) vs. Ex (-0.40)	2.40	no
ontology representation languages, abstract syntax, etc.	experience	Le (-0.22) vs. Ex (+0.23)	3.64	no

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

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

The overwhelming demand was for complying with common and established metaphors of user interaction. A quote from one participant sums

this potential source contributing to inefficiency: “*More standard compliance and consistency. The search works differently ... usual keyboard commands ... don’t always work...*”

In addition to the efficiency of the existing ontology management tools, two aspects were evaluated with respect to user experiences: (i) *usability* of the tool (which included accessibility and usefulness), and (ii) overall user *satisfaction* with the tool. The latter included comments regarding user interface intuitiveness, acceptability, customization, and so on.

As Table 2-4 shows, responses in this category are generally negative; participants considered the existing support as “very low” or “not very good.” Almost invariably, they were dissatisfied with the role of documentation, help system, tool tips, and various other tool-initiated hints. Support for tool customization—i.e. either its user interface or functionality—was also inadequate. A common justification of the low scores was (among others) the lack of opportunity to automate some actions, lack of support for keyboard-centric interaction, lack of support for more visual interactions. As can be seen from these examples, the reasons were quite diverse, and to some extent depended on the user’s preferred style.

Table 2-4. Selection of a few general observations across population

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

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

- *unclear error messages and hints* (e.g. red boundary around an incorrect axiom was mostly missed);

- *proprietary user interface conventions* (e.g. icons looked differently, search icon was not obvious, some menu labels were misleading);
- *lack of intuitiveness* (e.g. finding an operation, flagging a concept in the ontology so that it does not disappear, full- vs. random-text search);
- *inconsistent editing & amending of terms* (e.g. while “subClassOf” was visible at the top level of the editor, “equivalentTo” was hidden)

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

Measure/question	Type	Outcome	$\chi^2$	Sign.
level of overall satisfaction with the tools	tools	TB (+0.10) vs. Pr (−0.19)	2.67	no
overall satisfaction with tool's GUI environment	tools	TB (+0.10) vs. Pr (−0.24)	3.14	no
satisfaction with handling dependencies in ontologies	tools	TB (0.0) vs. Pr (−0.37)	7.65	yes
satisfaction with visualization and navigation support	tools	TB (−0.33) vs. Pr (−0.63)	6.00	yes
ease/speed of carrying out integrations	tools	TB (+0.50) vs. Pr (+0.10)	5.85	no
effort to get acquainted with the tool	experience	Be (−0.27) vs. Ex (+0.12)	3.02	no
satisfaction with support for interpreting inferences	experience	Le (0.0) vs. Ex (+0.07)	2.40	no
support for multiple ontology representation formats	experience	Le (−0.22) vs. Ex (+0.23)	3.64	no

As shown in Table 2-5, a significant difference of opinion was in the overall satisfaction with the tools, their design and intuitiveness, where it was more likely that people complained about Protégé than TopBraid. In this context, people tended to be more positive in the abstract than in the specific. Responses to specific queries were negative (between −0.500 and −0.100), yet overall experiences oscillate between −0.111 and +0.100. As we mentioned, the overall satisfaction with the TopBraid environment was more positive (some possible reasons were discussed above).

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

## 2.4 Lessons learned from the user study

Technology (such as OWL), no matter how good it is, does not guarantee that the application for its development would support users in the right tasks or that the user needs in performing tasks are taken on board. At a certain stage, each successful tool must balance the technology with user experience and functional features (Norman 1998). This paper explored some persevering issues with OWL engineering tools that reduce the appeal and adoption of otherwise successful (OWL) technology by the practitioners.

Although the tools made a great progress since the evaluations reported in section 2.1, issues with user interaction remain remarkably resilient. The effort was spent to make the formalisms more expressive and robust, yet they are not any easier to use, unless one is proficient in the low-level languages and frameworks (incl. DL in general and OWL's DL syntax in particular). Existing tools provide little help with the user-centric tasks — a classic example is visualization: There are many visualization techniques; most of them are variations of the same, low-level metaphor of a graph. And they are often too generic to be useful in the users' problems (e.g. seeing ontology dependencies or term occurrences in an ontology).

Table 2-6 highlights a few gaps between what the current tools provide and what people see as useful for framing problems in a more user-centric way. Some 'wishes' (white rows) already exist; e.g. Prompt (Noy and Musen 2003) for version comparison, but perhaps our findings may further improve design of the existing OWL engineering tools.

For instance, identification of frequently used operations and their correlations with errors and mistakes may provide us with opportunities to target the support towards most visible sources of user dissatisfaction. The most frequent steps in OWL development are the actual coding of definitions and import of ontologies (unsurprisingly), but, surprisingly, also search (71% users), re-conceptualization of restrictions and editing of logical expressions (both 54%), and locating terms in ontologies (46%). Compare these operations with the situations requiring assistance from facilitators (in Table 2-7).

Table 2-6. User attitudes to some functional features missing in existing tools (grey rows) and to some proposed extensions (white rows)

Current presence (grey) vs. wished-for feature	User attitude
Existing support for ontology re-use	-0.097 (not very good)
Support for partial re-use of ontologies	-0.739 (very poor)
→ flag chunks of ontologies or concept worked with	+0.519 (would be very useful)
→ hide selected (irrelevant?) parts of ontologies	+0.357 (would be useful)
Existing support for mappings, esp. with contextual boundaries	-0.065 (not very good)
Management and assistance with any mappings -0.480	(not very good / poor)
→ query ontology for items (instead search/browse)	+0.433 (would be useful)
→ compose testing queries to try out consequences of mappings	+0.045 (would be possibly useful)
Existing support for versioning, parallel versions/alternatives	-0.200 (not very good)
Existing visualizing capabilities & their adaptation	-0.536 (very poor)
→ mechanism to propagate changes between alternative versions	+0.519 (would be very useful)
→ compare/visualize different interpretations/versions	+0.700 (would be very useful)
→ visualize also on the level of ontologies (not just concepts)	+0.357 (would be useful)

Table 2-7. Observations of issues with OWL engineering and user interaction

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

One example we identified is the correlation between an incorrect logical conceptualization and confusion caused by ambiguous labels or dialogs. Other correlations were between problems with importing an ontology and absence or semantic ambiguity of appropriate widgets in the workspace, and between difficulties with definitions and the failure of tools to alert users about automatic syntactic checks (e.g. on brackets). The translation of a conceptual model of a restriction into DL-style formalism was a separate issue: 70% were observed to stumble during such definitions. From our data, we suggest considering multiple ways for defining and editing axioms (to a limited extent this partly exists in Protégé). Any way, DL may be good for reasoning, but it is by no means the preferred “medium for thinking” (even among ontology designers). This is not a novel finding, similar observations were made for other formalisms and their relationship to informal thought generation (Goel 1995).

Another issue is the gap between the language of users and language of tools; a high number of users was surprised by syntactically incorrect statements. In 64.3% sessions at least one issue due to syntax (e.g. of complex restrictions) was observed. Because of these minor issues they had to be alerted to by a facilitator, people tended to doubt results of other operations (e.g. search or classification) if these differed from what they expected. Lack of trust is problematic because it puts the tool solely in the role of a plain editor, which further reduces tool's initiative. In an attempt to restore 'user trust,' some tools (e.g. SWOOP) move towards trying to justify their results (Kalyanpur, Parsia *et al.* 2005).

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

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

### 3. USER INTERACTION WITH ONTOLOGIES

In the previous section we mostly considered one particular category of the users with respect to ontologies; namely, those users who want to author, design and amend ontologies as a part of some integrative task. This is an important group of users; however, these are not necessarily the only users who may have a need to interact with networked ontologies. The issue of interacting with ontologies effectively and efficiently is much more pressing with less experienced users, who carry out an ad-hoc, occasional ontology-related task — as shown, to some extent by our study reported in section 2.

Therefore, in this section we explore the problem of user interaction with ontologies more in depth, from several angles.



### 3.1 Configurable user interfaces

One of the findings in the user study we briefly described in section 2.3 was pointing to the fact that the ontology engineering environments tend to be reasonably modular, but they are essentially built alongside “one size fits all” strategy. In reality, such a strategy is rare among the successful software products. As users within the corporate intranets or outside of companies take on different roles, they come across and emphasize different business needs from, in principle, the same information content. Subsequently, they typically expect the tools of their trade would somehow reflect those different business needs.

One of the most often mentioned features of a new software product is an easy customization of its user-facing components. We explore this theme in the second half of the chapter on HCI challenges in ontology engineering. The quote from a software company’s catalogue (anonymized by the authors) below summarizes the point:

*[Our product] provides an easy to configure user interface enabling you to meet diverse business needs across your enterprise, as well as support localization. [Among other functionalities, the product supports] menu localization and support for international languages, enabling and disabling functions for users based on their permissions, [...]*

Users involved in ontology-driven production of information and knowledge need to be equipped with a range of software configurations and diverse user interfaces to deliver the outcomes of their work as effectively and efficiently as possible. There are two broad strategies how one can match the tools to the needs:

1. different tools for different users and different purposes;
2. different configurations of one tool or toolkit for different users or purposes.

The two strategies are not mutually exclusive; very often we find that users rely on a limited range of tools, and then may have different, specialized configurations for some of those tools. Let us briefly consider the key advantages and disadvantages of the above approaches: In the former situation, tools are well defined but apparently independent of each other. This may lead to a proliferation of a large number of highly specialized tools, something that is overwhelming and unlikely to alleviate the user’s confusion. Moreover, with specialized tools, there is an increasing risk of them being mutually less compatible or compatible on a rather cumbersome level (e.g. import/export mechanism of various graphical editors is a good example of this compatibility issue). The main advantage is that the user will

only get to work with tools and interfaces s/he necessarily needs to carry out given tasks, and nothing more.

In the latter situation, we tend to see more complex and multi-functional tools that can exhibit a variety of user interfaces and user interaction components in different situations. In many tools of this type, we see an aggregation of functionalities and a fairly seamless switching between many tasks the user may carry out at some point. This is essentially a “one-stop shop” approach where the user has (almost) everything they may ever need already inside the tool, and only needs to activate different configurations. A typical example of this would be editors like Microsoft Word, and its ‘rich document editor’ face, as opposed to (say) ‘content revision’ face or ‘mail merge and distribution’ face.

Formally, these notions were explored by Shneiderman (2000) who introduced so-called *universal usability*. While this rather broad issue is clearly beyond the scope of this chapter, Shneiderman points to several factors that may affect the tool usability. These are factors that vary from one user to another, and hence trigger a degree of adaptation to the user interface. Importantly, Shneiderman highlights many common factors that are not always recognized as valid reasons for UI customization. For example, he talks about technological variety (i.e. the need to support a range of software and hardware platforms, networks, etc.), about gaps in user knowledge (what users know, what they should know, etc.), or about demographic differences (skills, literacy, income) or environmental effects (light, noise, etc.)

One approach to achieving more universal usability of a tool is to introduce user interface adaptation into the loop. The rationale is that while a standard UI may not fit the user completely, it might be tweaked so that it gets as closely as possible to the user needs. There are two distinct strategies of how UI adaptation may be accomplished. Since this differentiation may have impact on what is actually modified in the tool, we decided to include this brief detour to generic issues of adaptation. The two strategies distinguish between the following types (Kules 2000):

- **adaptive UI:** These are systems and user interfaces that are capable of monitoring its users, their activity patterns, and automatically adjust the user interface or content to accommodate these local differences in activity patterns (which may be due to user’s skill, preference, etc.).
- **adaptable UI:** These are systems and user interfaces that allow the users to control and specify adjustments, and often come with the provision of some guidance or help.

According to the informal definitions, the difference is in the actor; who performs the adaptation act. In adaptive UI-s it is the tool, applications or the

system that takes the active role; whereas in adaptable UI-s it is the human — typically the actual user of the system, but possibly another user (such as system administrator).

Why do we mention user interface adaptation in this context? Ontologies are highly structured, formalized artefacts that have sufficient expressiveness to describe the structure of a system, tool, or its user interface. Considering that such common tools as Web browsers make use of ontological formalisms to support customization and thus make life easier for the user, it is rather surprising that very little of a similar approach is used to improve the tools for interacting with ontologies.

## 4. USERS AND ONTOLOGY ENGINEERING

In this section we briefly sketch some of the existing approaches that have been developed mostly in the context of personalization and scalability (i.e. the capability to work with large data sets). This overview is intended to be informative rather than exhaustive; it is intentionally compiled on a level that abstracts from individual tools and method to approaches and strategies.

As ontologies become more and more complex and as they are integrated into networks of ontologies, it is reasonable to investigate the means, which would be capable of making a large network of complex ontologies more manageable. The customization and personalization of ontologies includes, in principle, two areas relevant to ontologies:

- customization of the view on an ontology, e.g. during exploring a network of ontologies. This customization is more or less ad-hoc and the results of the customization may be discarded once the user proceeds with exploring the ontology. This customization during exploring an ontology tries to reduce the complexity of an ontology and only shows parts which are relevant for the current user.
- customization for the purposes of reusing ontologies and integrating them into a network with other ontologies according to specific needs (e.g. during the ontology deployment, reasoning or design phases). Here the results of the customization will often be integrated into the edited ontology.

As one basis for the customization, we analyze and briefly overview user profiles and profiling work, followed by techniques for exploring and navigating in large data sets (including ontologies), and finally we touch on the role of algebraic operators to manipulate the topology or content of ontologies.

## 4.1 User profiling

User profiles are seen here as a way to describe some user properties or characteristics and thus as a representation of the context of a user. Such a profile may for example provide information about the role of a user, the domain of interest or the current task. This information about the context helps in a user-tailored information delivery, e.g. by offering personalized ontology views. When talking about the user, it is important to mention that we can decide to have an *abstract user*—this would be, in principle, corresponding to any member of a group of users in a particular situation.

A user profile can be constructed in different ways depending on the data it includes and the methods used for its construction, including manual, semi-automatic and automatic methods. Each of them has some advantages and disadvantages. For a review of specific user profile acquisition techniques, consider e.g. sources mentioned in (Dellschaft, Dzbor *et al.* 2006). Let us focus in this chapter on how such profiles might be deployed and used in the context of ontology management.

In principle we see the role of user profiles as twofold: (i) as a means allowing recommendations based on some typicality effects, and (ii) as a means having a predefined description on the actions to be applied by the system, depending on some predefined user profile characteristic.

In the former case, it is interesting to acquire information, e.g. about which ontology views a given category of users prefers, what level of detail they use in annotating documents using that ontology, or which partition of a larger ontology they mostly interact with (and for what purpose).

In the latter case, a user profile may act as a kind of task definition for the activity the user is expected to carry out—an example of such a situation might be provision of an ontology view that would be less suitable to editors but much more efficient to validators.

There are many profiling systems in existence; most of them developed in the context of user interaction with Web documents and Web browsing. One example is Lifestyle finder (Krulwich 1997)—a collaborative recommendation system as it groups similar users based on the similarity of their manually constructed user profiles. It recommends potentially interesting Web documents to the user based on the ratings of the documents provided by similar users. A similar example is NewsWeeder (Lang 1995), a system for electronic Usenet news alerts.

An example of the semi-automatic approach is OntoGen (Fortuna, Mladenic *et al.* 2006) that constructs a profile from a set of documents provided by the user, and then proposes a topic hierarchy (i.e. a simple ontological model of the user's interests) that can then be used e.g. to recommend navigational steps to the user (Fortuna, Mladenic *et al.* 2006) or

to visualize a particular collection based on the hierarchy of user interests (Grcar, Mladenec *et al.* 2005).

User profiling is one of the important aspects for customizing human-ontology interaction. User profiles can be used to mesh different data sources, where the preferences for a data source are based on the user profile (initially manually, but possibly adjusted based on the user's activity). User profiling can also be used for providing a personalized view on an ontology based on the ontologies previously constructed by the same or a similar user. Such a personalized view can be seen as putting ontologies in a particular context, which is familiar to the user (and hence, simplifies his or her interpretation of the ontology).

## 4.2 Navigating in complex conceptual structures

Since ontologies are often formal artefacts, they need some transformation to be comprehensible to the ordinary users. This is rarely straightforward. First, ontological datasets are relatively large; they contain thousands of statements the user may need to interact with. For example, a fairly simple geographic ontology of regions in New York state<sup>5</sup> contains as many as 59,000 unique statements just about congressional districts in a single US state. Second, ontologies could be complex structures representing different types of relationships. If each of such potential relations is treated as a dimension in which allowed values could be depicted, then even a moderately complex ontology leads to a multi-dimensional space, which poses challenges for navigation and interaction — in particular, when human cognition naturally prefers (and is able of coping with) two or three dimensions.

Two strategies that may apply to ontologies are their *reduction* and *projection*. Where reduction is concerned with showing *less at a given point in time* (in our case, fewer concepts, entities or relationships), *projection* works by showing the same set of concepts, entities and relations differently. The two strategies are somewhat complementary.

### 4.2.1 Reducing complexity of navigation

One common reduction strategy has been implemented in a number of faceted browsers (but not in the context of ontologies). The key principle of this strategy is that large collections (e.g. libraries or galleries) have many dimensions according to which they can be viewed, browsed, searched or navigated. Thus, faceted navigation is an interaction style whereby users

---

<sup>5</sup> A serialization and a downloadable version of this ontology is available from:  
<http://www.daml.org/2003/02/fips55/NY.owl>

filter an appropriate set of items by progressively, step-by-step selecting from valid dimensions of a particular classification. That classification can be created according to many principles (including ontology-derived).

Earlier representatives of this strategy include Flamenco—a portal for browsing fine arts collections (Hearst 2000; Yee, Swearingen *et al.* 2003) or mSpace—an access site to a repository about the computer science in the UK (Schraefel, Karam *et al.* 2003). More recent examples include e.g. Longwell and Fresnel (Pietriga, Bizer *et al.* 2006) from MIT's Simile project as representatives of generic frameworks and vocabularies (respectively) for faceted navigation through RDF collections and for specifying facets. Other recent examples include BrowseRDF (Oren, Delbru *et al.* 2006), a generic RDF browser, or /facet (Hildebrand, van Ossenbruggen *et al.* 2006), an RDF browser used in a manner similar to Flamenco, but in the context of the Dutch cultural heritage project. Nonetheless, most of the above tools focus on data rather than triple-level graph structures typical for ontological concepts and relations.

User interaction in faceted style usually starts with an overview of the browsed collection, which often yields a large number of possibly relevant matches. In the subsequent browsing steps, this 'relevant' set is structured according to selected categories (e.g. locations, styles, themes, etc.). Alternatively, the user may narrow the view down by referring to hierarchical classification (if available). The navigation may end with accessing a particular item from the collection. We use term 'may' because alongside the item the user always sees all other categories and metadata that provide bridges to alternative collections.

A slightly different view on the principle of faceted navigation is advocated by the authors of CS AKTive Space and the family of similar mSpace-based applications (Shadbolt, Gibbins *et al.* 2004). The faceted views for browsing the collections are fairly typical, but there is one pane that also uses a projection strategy—geographic data are shown naturally, i.e. on a map. A useful side effect of such projections is that they enable the user to express relations very succinctly (including fuzzy ones such as *near* or *in the South*). Unlike Flamenco, mSpace is more tightly linked to ontologies—they act as the primary classification of different facets that are available to the user.

To explore the role of spatial metaphors in navigating complex structures we point e.g. to work by Mancini (2005), who experimented with ways how the same content may yield different interpretation if presented (and navigated) in a spatially different manner. Nevertheless, the use of such techniques for ontology management needs further research, before we are able to link them to particular use case scenarios and requirements. More details on how faceted browsers may assist ontology management has been

provided in (Dellschaft, Dzbor *et al.* 2006), which also formed the base for this section.

In general, what faceted browsers like Flamenco support rather well is the iterative formulation of the search queries or navigational goals. Key advantage of this technology is the step away from forcing the user to go through deep, complex hierarchies in order to find items they are interested in. Users only navigate to the next slice by following some conceptual clues (e.g. sub-categories or orthogonal views). Arguably, faceted navigation seems to be a more natural way of coping with messy, conceptually complex space, than a rigid, hierarchical tree-like structure.

Thus, the “divide and conquer” strategy also works in the context of complex conceptual spaces such as ontologies. What is hard to visualize at once because of variability and differences between different relationships, can be split into sequences of partial visualizations through which it is easier to move and which are also more comprehensible to the end user. On the other hand, faceted browsers suffer from the scaling issue; i.e. they work reasonably well with a few well-defined facets that can be arbitrarily combined by the end user. For instance, CS AKTive Space used only three key (i.e. navigable) dimensions (location, topic and institution). In Longwell, deployed for MIT OpenCourseWare, there are similarly three dimensions (level of study, teacher and keywords). An ongoing tension emerges between offering as many facets to the user as possible while simultaneously helping to reduce navigational complexity.

#### 4.2.2 Projections for large ontological data sets

In addition to conceptual and relational complexity that has been tackled by the research into faceted navigation, another similarly hard task is to navigate through large datasets. A number of projections were proposed to tackle this. In particular, the fish-eye metaphor enables customizable navigation; it uses different properties of the objects in a knowledge base to create clusters of different granularity and of different semantics. For example, Komzak and Slavik (2003) illustrate this capability to handle large networks of diverse but conceptually related data in the context of visualizing the 200k strong student population of The Open University in the UK, which can be shown on a geographic, per-faculty, per-program or per-course basis.

The strategy relies on showing the *contextual fringe* of a part of the semantic network not corresponding to a particular user’s query or intention using more coarse-grained clusters than the part that actually corresponds to the query and is currently *in focus*. The authors also open up the context-focus metaphor (Lamping, Rao *et al.* 1995), so that each particular focus

(fine-grained view) can be embedded into an arbitrary context (coarse-grained view).

Another algorithm based on the focus-context metaphor is SpaceTree (Plaisant, Grosjean *et al.* 2002). SpaceTree is a tree browser to some extent similar to hyper trees (Lamping, Rao *et al.* 1995). It addresses one difficulty of the hyperbolic geometry; namely constant updating of the visual representation, which makes it hard for the user to create a mental map of the ontology, hierarchy or taxonomy. SpaceTree uses dynamic rescaling of tree branches to fit within a constrained space; miniature tree icons are used to indicate the depth, breadth and size of the sub-trees hidden behind a given node.

A different example for projecting ontologies is provided by the “crop circles” metaphor (Parsia, Wang *et al.* 2005). As with the fish-eye, this metaphor also shows some implicit topography in an overview mode. In CropCircles classes and partitions are represented as circles. One can hover over a particular node in the visualization to see the class it actually represents. By clicking on a class one can quickly highlight its immediate neighborhood (children, parents). Also, zooming in and out is easily supported in this view, and as the recent study from Wang and Parsia (2006) showed, the metaphor in some cases could outperform other visual techniques (especially in the context of viewing richly interlinked and deep ontologies).

On a more traditional level, ontologies are often perceived by many developers, researchers and users as predominantly hierarchies of subsumed concepts; i.e. structures where one concept is a kind of another concept (as in “Ford is a Car”). Hence a lot of effort was put into navigating these, so-called *isA* structures. Techniques like IsaViz<sup>6</sup> focus on the structurally dominant relationship in any ontology (subClassOf). Two key shortcomings of this approach are: (i) its usefulness rapidly falls with the depth of a hierarchy, and (ii) very few graphs actually have a neat hierarchical structure. The *isA* graphs make visually interesting demonstrations, but by definition, they do not contain various lateral or horizontal relations (Brusilovsky and Rizzo 2002).

Some of the more recent developments in the field of ontology visualization took an approach more centered on the user needs. A good example of this is Jambalaya (Ernst, Storey *et al.* 2003), a project that started with the aim to visualize rich ontology graphs and was initially driven by the technological needs. However, at the application re-design stage, the needs of real users were considered for particular audiences comprising the biologists in a large national research center. These requirements came from observing the actual users — biologists, and conjecturing potentially useful

---

<sup>6</sup> More information available from <http://www.w3.org/2001/11/IsaViz>



functional requirements from these observations. As a result, Jambalaya is more balanced in addressing a range of users needs on an ontology visualization package.

One the level of underlying technology, Jambalaya's visualization is still based on the metaphor of a graph, but allows more customization of what can be visually depicted. Particularly its FilmStrip metaphor (Ernst, Storey *et al.* 2003) suggests an interesting compromise between data overviews and its specific context. Yet, due to realizing this idea through showing the relevant information as nodes, the outcome is full of boxes and overlapping edges. These often achieve the opposite of a positive user experience, as the overlapping graph sub-structures may easily obscure much of the underlying semantic structure.

Many practical ontologies use a range of relationship; e.g. UK Ordnance Survey reports on their use of a range of ontological relationships that may easily create issues if inappropriately visualized (Dolbear, Hart *et al.* 2006). In particular, they highlight issues with fairly common geo-spatial relationships like *contained within*, *next to* or *surrounded by*. In each of the cases illustrated, merely showing two nodes from the low-level data representation linked with a declared or inferred labeled arc is not of much use. For instance, in some cases objects such as fields may be both *surrounded by* and *contained within* and be *inside* of a wall. However, if field F is *contained within* something else (e.g. wall), by definition *it cannot be next to* another field F,' since they would need to share the 'container.' However, to anybody visualizing a dataset containing fields F and F' it makes perfect sense to 'ignore' the dividing walls and talk just about the fields.

#### 4.2.3 Benefits of navigational and visualization techniques

Cognitive studies, one of the recent examples is a study by Demian and Fruchter (2004), show that there are several mutually not fully compatible requirements on interacting through visual user interfaces:

- a need to find a particular item (e.g. knowing some of its properties),
- a need to explore the context in which an item is defined (e.g. what does it mean if we say that "Ford is a Car"), and
- a need to establish the difference between two or more items, which may include temporal differences due to evolution or various conceptual differences (e.g. "Ford Transit is a Ford, but not a Car, and this is because...")

The simple IsaViz and related techniques basically address only the second need identified above, and even that to a very small extent. The

implications of the discussion in the above paragraphs are that there is unlikely to be one perfect method or technique for scaling up the navigation through structured datasets. What is more likely to work is reusing familiar metaphors, such as the FishEye projections or CropCircles. However, it seems equally important to use these metaphors at the right point during the process of navigating ontologies. Crop Circles, for instance, seem to fit best if one is interested in seeing broad relationships among several ontologies. Map-like FishEye projections, on the other hand, seem to show a finer level of granularity—e.g. when one wants to explore the relationship of networked ontologies to a particular concept in one ontology.

One approach that has not been mentioned so far, but which actually could combine the need of dealing with large-scale datasets with the need to simplify the ontological definitions, is inspired by maps and mapping metaphor. By definition, any map is essentially a projection of a particular world (most often a landscape) onto a paper (or screen). One can imagine creating such domain landscapes from several different perspectives. For instance, a landscape of research topics in Europe is likely to look somewhat differently from the landscape of UK's football or the landscape of great maritime voyages.

Assume we have several pre-computed landscapes available that show the key terms of a particular domain (an example is shown in Figure 2-1), their links, relationships, closeness, etc. When we take one or several ontologies, we can cover these domains with the given ontologies. In some cases, the coverage would be better and more precise than in others. Different ontologies would be positioned into different regions of the landscape—dependent on which landscape the user takes as a foundation for his or her navigation. Although we have given this example with ontologies in general, most of the current tools deal only with data (possibly annotated using ontologies). Hence, adaptations of the familiar techniques are needed to apply to ontologies as topological structures, not only as data sets.

Another interesting strategy is motivated by work done by Collins, Mulholland *et al.* (2005) on spotlight browsing. The principle of this navigation strategy is again based on a metaphor—a torch throwing a beam of light. The user selects a resource or a concept from a particular collection; then the collection is dynamically restructured so that it conveys interesting properties, clusters, etc. that may be relevant to the initial 'spot.' These additional items and concepts are then structured around the original spot by calculating their semantic closeness. The navigation is then equivalent to shedding a light beam (as shown in the mockup in Figure 2-1), which puts certain concepts into light (i.e. into navigable focus) and certain other items into shadow (i.e. into non-navigable periphery).

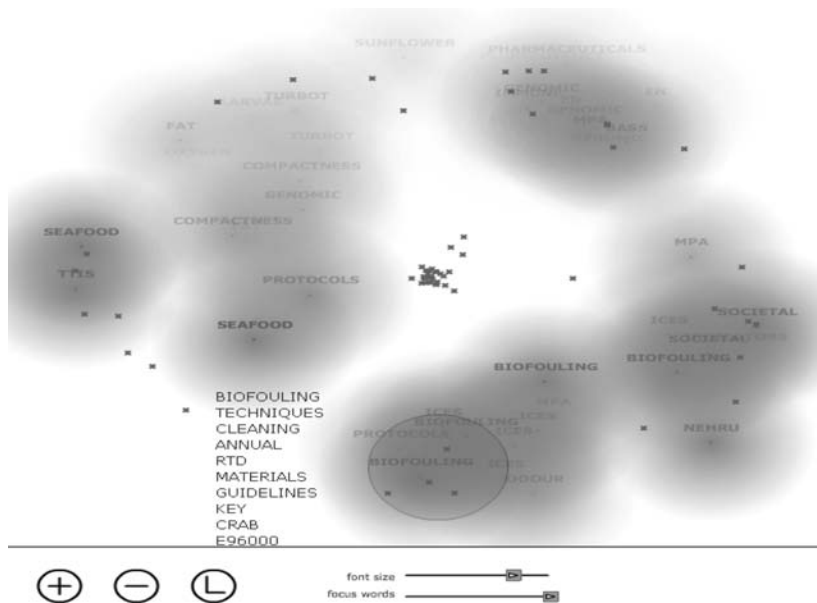


Figure 2-1. Mock-up of a 2D rendered landscape with two ontologies broadly covering and mapping different sections of it. Green areas roughly correspond to different ontologies and red crosses to selected terms whose distance/mutual positions depend on a particular corpus.

### 4.3 Customizing ontologies

One of the early works toward ontology customization came from Mitra and Wiederhold (2004), who proposed a modularized approach to creating ontologies as this would ease ontology reuse and would help breakdown the required effort into smaller, manageable pieces. To that goal, they describe a general idea of ontology customization operators that would support such a modularized approach and help combine the modules to larger ontologies. Examples of their operations include, e.g.:

- selection from an ontology (there are different criteria for this);
- intersection of several ontologies (i.e. a common denominator);
- union or extension of several ontologies;
- differentiation or discrimination between ontologies, etc.

In addition to the binary or n-ary operations, there is an important set of unary operations, those working on a single ontology. It is this particular set that is of interest in the context of our objective discussed in this chapter. For

example, work by Jannink, Mitra *et al.* (1999) describes four binary and four unary operators. Among them are some interesting unary operators:

- *summarize* — centralizes the ontology into groups of similar concepts;
- *glossarize* — lists terms subordinate to a given concept without any of the recognition of the sub-structure;
- *filter* — extracts instances from ontology according to a given predicate;
- *extract* — reduces concepts and the possibly corresponding instances from the ontology according to a given predicate/condition.

Particularly useful operations, from the perspective of reducing ontology complexity, are the first two operations: summarization and glossarization. Both essentially drawing on the latter two operations, but providing useful interpretative viewpoints on a complex conceptual structure. In this chapter we are not going into more depth with regard to customization operations and how they may be realized, a brief overview of some tools and their support for this task is discussed, for instance, by Dellschaft, Dzbor *et al.* (2006).

Nonetheless, let us at least mention how the operators mentioned above might be related to section 4.1 (user profiles) and section 4.2 (ontology navigation). In both previous sections we relied on the fact that a part of the ontology is known, but we haven't really said how such parts might be obtained. For example, for the spotlight or fish-eye facility, we may need a central, in-focus portion of an ontology together with several summaries of the surrounding contextual fringes.

These requirements may be directly linked to the aforementioned operations for ontology customization — extraction (to get a focus area) and summarization (to obtain meaningful but brief summaries of what lies around the focal point). Hence, in general, the techniques described in this section may be seen as data feeds for the purpose of visualization and navigation methods, which in turn may act as points where the user may make choices, which could be captured in a specific profile.

Next we shall present how the three apparently independent areas may relate together in a kind of user support “stack.”

## 4.4 Illustrative scenario — putting it all together

Imagine we work with several ontologies, which we want to navigate. Among others we have FishBase, AgroVoc, FIGIS, and other ontologies typically used by agricultural experts<sup>7</sup>. Let us assume our expert wants to edit parts of the ontology related to *Albacore* tuna. These need to be located,

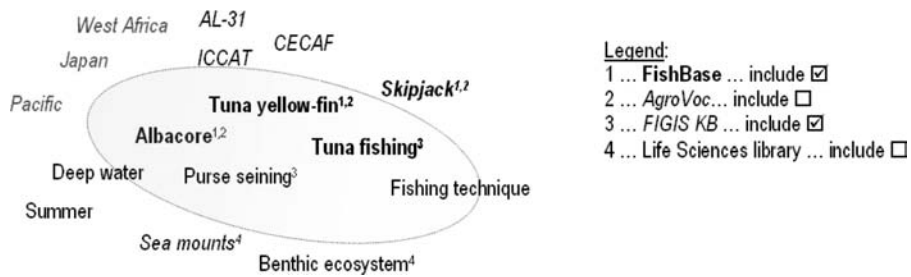
---

<sup>7</sup> To learn more about these ontologies visit <http://www.fao.org/fi>

extracted, and presented appropriately, because the number of related terms is potentially exponentially large.

First, large ontologies may be reduced so that they contain the minimal number of concepts surrounding the albacore tuna, which are still ontologically complete and sound. This may be achieved by applying one of the ontology reduction/extraction operators mentioned in section 4.3. The extraction may find overlaps and possibly generalizations of term subsets, so that the diversity could be expressed using a smaller number of concepts.

Different alternative navigational paths can then be visually summarized in a manner following *Figure 2-1*. The initial position of the yellow “light beam” would reflect that exploratory path through the concept cloud that seems to be best covered by the existing fishery ontologies. The numbers in superscript in the figure may e.g. refer directly to the internal formal resources referring to a particular theme (e.g. FIGIS, AgroVoc, etc.). In addition, the weight of the terms is given by their ontological reliability and provenance—where our expert may quickly see that the fish species are particularly well conceptualized.



*Figure 2-1.* Mock-up of an ontology summary view showing concepts related to the focal term (*Albacore*) and ontologies covering these terms. Typefaces may reflect e.g. trustworthiness of terms against ontologies with same italic/bold typeface on the right.

In the shape as shown in *Figure 2-1*, an expert may easily see different dimensions corresponding to diverse ontological relationships around the concept of albacore tuna. Such a conceptual summary space may be easily reorganized without too much cognitive overhead on the part of our expert. For instance, re-pointing the beam towards the red section (which may denote some ontological inconsistency), it is possible to rapidly refine a particular type of ontological relationship. In our case, assume we target the locality and fish habitat relations. An outcome of such an action is sketched in *Figure 2-2*, where one sees more relevant ontologies, different concepts emerging in focus, and others fading into the fringe.

Thus, a typical use case applying the three layers of user-centred ontology management we discussed in this section, presents a mesh of several familiar techniques. The three areas we mentioned — user profiling, navigation and visualization techniques, and customization operators — can be seen as three layers of a stack, which influence each other in a variety of ways. For example, based on a user profile, one may prefer a particular navigational technique; such a technique may need to draw upon a specific customization operation. That, in turn, may help keep the profile up to date, etc. Hence, the three layers addressing complex user issues in our illustrative scenario are manifested in the following ways:

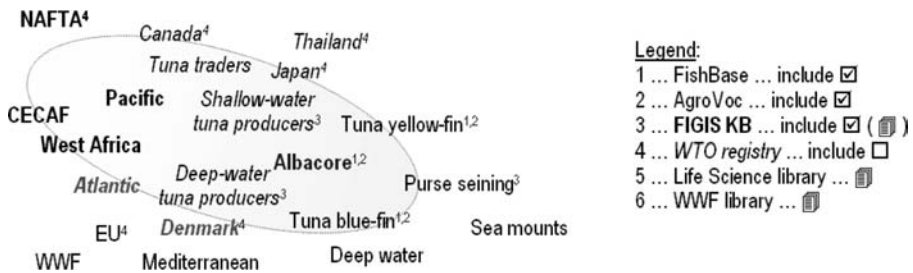


Figure 2-2. Mock-up of the repositioned focus of related terms and ontologies covering these terms

- User profiling techniques:
  - acquiring user and group profiles;
  - using machine learning to manage user profiles;
- Customized, abstract-level interaction with ontologies:
  - hiding the low-level aspects of several ontology engineering tasks;
  - making sense of links and relations within/between ontologies;
  - ontology visualization on the level of domain coverage;
  - spotlight browsing and other less common browsing extensions;
- Ontology customization operations:
  - reducing ontology complexity;
  - modularization and view customization based on user-selected criteria;
  - customization operations such as module reduction, compounding, differencing, etc.

## 5. CONCLUSIONS

In this chapter we briefly covered the broad and rather complex area of human-ontology interaction. We started with reviewing generic tenets of HCI and their relevance to ontology management. We then presented some empirical evidence highlighting the fact that the existing ontology engineering tools are still at a very early developmental stage (from the software lifecycle point of view). We concluded this part with highlighting several functional opportunities that seem to be missing in the existing tools for ontology management, in particular for ontology engineering.

Then we offered an exploratory survey of some areas that are not commonly associated with ontological engineering, and considered what roles these techniques may play in making the human-ontology interaction more mainstream and more acceptable for so-called ordinary users. In particular, we started with user profiling, elaborated on the use of data visualization, navigation and exploration techniques, and briefly touched on the need to investigate ontology customization operations and methods, as the foundation of our triple stack of technologies that may make life of the user easier.

## ADDITIONAL READING

- Collins, T., Mulholland, P., *et al.* (2005). *Semantic Browsing of Digital Collections*. Proc. of the 4th Intl. Semantic Web Conf., Ireland, pp.127–141.
- Dellschaft, K., Dzbor, M., *et al.* (2006). Review of methods and models for customizing/personalizing ontologies, NeOn project: From [http://neon-project.org/web-content/index.php?option=com\\_weblinks&catid=17&Itemid=35](http://neon-project.org/web-content/index.php?option=com_weblinks&catid=17&Itemid=35) (April 2007).
- Duineveld, A. J., Stoter, R., *et al.* (2000). “WonderTools? A comparative study of ontological engineering tools.” *Intl. Journal of Human-Computer Studies* **52**(6): pp.1111–1133.
- Dzbor, M., Motta, E., *et al.* (2006). *Developing ontologies in OWL: An observational study*. OWL:Experiences & Directions wksp., Georgia, US.
- Ernst, N. A., Storey, M. A., *et al.* (2003). *Addressing cognitive issues in knowledge engineering with Jambalaya*. Knowledge Capture Conference (K-Cap), Florida, US.
- Norman, D. (1998). *The Invisible Computer*. Cambridge, MA, MIT Press.
- Shneiderman, B. (2000). “Universal Usability: pushing human-computer interaction research to empower every citizen.” *Communications of the ACM* **43**(5): pp.84–91.
- Shneiderman, B. and Plaisant, C. (2004). *Designing the User Interface: Strategies for effective human-computer interaction*, Addison-Wesley, 672 pages.
- Storey, M. A., Lintern, R., *et al.* (2004). *Visualization and Protégé*. 7th International Protégé Conference, Maryland, US.

## REFERENCES

- Brusilovsky, P. and Rizzo, R. (2002). "Map-Based Horizontal Navigation in Educational Hypertext." *Journal of Digital Information* **3**(1): pp.156.
- Collins, T., Mulholland, P., *et al.* (2005). *Semantic Browsing of Digital Collections*. Proc. of the 4th Intl. Semantic Web Conf., Ireland, pp.127–141.
- Colman, A. M. (2001). *A Dictionary of Psychology*. Oxford, Oxford University press, 864 pages.
- Dellschaft, K., Dzbor, M., *et al.* (2006). Review of methods and models for customizing/personalizing ontologies, NeOn project: From [http://www.neon-project.org/web-content/index.php?option=com\\_weblinks&catid=17&Itemid=35](http://www.neon-project.org/web-content/index.php?option=com_weblinks&catid=17&Itemid=35) (April 2007).
- Demian, P. and Fruchter, R. (2004). CoMem: Evaluating Interaction Metaphors for Knowledge Reuse from a Corporate Memory. Stanford, Center for Integrated Facility Engineering, Stanford University: 47 pages.
- Dolbear, C., Hart, G., *et al.* (2006). *What OWL has done for Geography and why we Don't Need it for Map Reading*. OWL:Experiences & Directions workshop, Georgia, US.
- Duineveld, A. J., Stoter, R., *et al.* (2000). "WonderTools? A comparative study of ontological engineering tools." *Intl. Journal of Human-Computer Studies* **52**(6): pp.1111–1133.
- Dzbor, M., Motta, E., *et al.* (2006). *Developing ontologies in OWL: An observational study*. OWL:Experiences & Directions wksp., Georgia, US.
- Ernst, N. A., Storey, M. A., *et al.* (2003). *Addressing cognitive issues in knowledge engineering with Jambalaya*. Knowledge Capture Conference (K-Cap), Florida, US.
- Fensel, D. and Gómez-Pérez, A. (2002). A survey on ontology tools, OntoWeb Project: [http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/OntoWeb\\_Del\\_1-3.pdf](http://www.aifb.uni-karlsruhe.de/WBS/ysu/publications/OntoWeb_Del_1-3.pdf) (April 2007).
- Fortuna, B., Mladenic, D., *et al.* (2006). *Semi-automatic data-driven ontology construction system*. Proc. of the 9th Multiconference on Information Society, pp.223–226.
- Goel, V. (1995). *The Sketches of Thought*. Massachusetts, US, MIT Press, 274 pages.
- Grcar, M., Mladenic, D., *et al.* (2005). *User profiling for interest-focused browsing history*. Proc. of the 8th Multiconference on Information Society, pp.223–226.
- Gruber, T. R. (1993a). "A Translation approach to Portable Ontology Specifications." *Knowledge Acquisition* **5**(2): pp.199–221.
- Gruber, T. R. (1993b). "Towards principles for the design of ontologies used for knowledge sharing." *Intl. Journal of Human-Computer Studies* **43**(5/6): pp.907–928.
- Hearst, M. (2000). "Next Generation Web Search: Setting Our Sites." *IEEE Data Engineering Bulletin* (Special issue on Next Generation Web Search, Luis Gravano (Ed.)).
- Hildebrand, M., van Ossenbruggen, J., *et al.* (2006). *Ifacet: A browser for heterogeneous semantic web repositories*. Proc. of the 5th Intl. Semantic Web Conf., Georgia, US, pp.272–285.
- Jannink, J., Mitra, P., *et al.* (1999). *An algebra for semantic interoperation of semistructured data*. IEEE Knowledge and Data Engineering Exchange Workshop, Illinois, US.
- Kalyanpur, A., Parsia, B., *et al.* (2005). "Debugging Unsatisfiable Classes in OWL Ontologies." *Journal of Web Semantics* **3**(4).
- Kirkpatrick, D. L. (1994). *Evaluating Training Programs: the Four Levels*. San Francisco, Berrett-Koehler Publishers, 289 pages.
- Komzak, J. and Slavik, P. (2003). *Scaleable GIS Data Transmission and Visualisation*. Intl. Conf. on Information Visualization (IV), UK.
- Krulwich, B. (1997). "Lifestyle finder." *AI magazine* **18**(2): pp.37–46.



- Kules, B. (2000). "User modeling for adaptive and adaptable software systems." *UUGuide: Practical design guidelines for Universal Usability*, From <http://www.otal.umd.edu/UUGuide> (April 2007).
- Lamping, J., Rao, R., et al. (1995). *A focus-context technique based on hyperbolic geometry for visualizing large hierarchies*. Proc. of the Conf. on Human Factors in Computing Systems.
- Lang, K. (1995). *News weeder : Learning to filter netnews*. Proc. of the 12th Intl. Conf. on Machine Learning.
- Mancini, C. (2005). *Cinematic Hypertext: Investigating a New Paradigm*. Amsterdam, The Netherlands, IOS Press, 192 pages.
- Mitra, P. and Wiederhold, G. (2004). An ontology composition algebra. *Handbook on Ontologies*. S. Staab and R. Studer. Heidelberg, Germany, Springer Verlag: pp.93–113.
- Norman, D. (1998). *The Invisible Computer*. Cambridge, MA, MIT Press.
- Noy, N. F. and Musen, M. A. (2003). "The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping." *International Journal of Human-Computer Studies* **59**(6): pp.983–1024.
- Noy, N. F., Sintek, M., et al. (2001). "Creating Semantic Web Contents with Protege 2000." *IEEE Intelligent Systems* **16**(2): pp. 60–71.
- Oren, E., Delbru, R., et al. (2006). *Extending faceted navigation for RDF data*. Proc. of the 5th Intl. Semantic Web Conf., Georgia, US, pp.559–572.
- Parsia, B., Wang, T., et al. (2005). *Visualizing Web Ontologies with CropCircles*. Proceedings of the ISWC 2005 Workshop on End User Semantic Web Interaction, Ireland.
- Pietriga, E., Bizer, C., et al. (2006). *Fresnel: A browser-independent presentation vocabulary for RDF*. Proc. of the 5th Intl. Semantic Web Conf., Georgia, US, pp.158–171.
- Pinto, S., Peralta, N., et al. (2002). *Using Protégé-2000 in Reuse Processes*. Evaluation of ontology-based tools (EON), pp.15–25.
- Plaisant, C., Grosjean, J., et al. (2002). *Spacetree: Supporting exploration in large node link tree, design evolution and empirical evaluation*. Proc. of the Intl. Symposium on Information Visualization.
- Shraefel, M C, Karam, M., et al. (2003). *mSpace: interaction design for user-determined, adaptable domain exploration in hypermedia*. Workshop on Adaptive Hypermedia and Adaptive Web Based Systems, UK, pp.217–235.
- Scriven, M. (1991). Beyond Formative and Summative Evaluation. *Evaluation and Education: A Quarter Century*. M. W. McLaughlin and D. C. Phillips. Chicago, University of Chicago Press: pp.19–64.
- Shadbolt, N. R., Gibbins, N., et al. (2004). "CS AKTive Space: or how we learned to stop worrying and love the Semantic Web." *IEEE Intelligent Systems* **19**(3): pp.41–47.
- Shneiderman, B. (2000). "Universal Usability: pushing human-computer interaction research to empower every citizen." *Communications of the ACM* **43**(5): pp.84–91.
- Shneiderman, B. and Plaisant, C. (2004). *Designing the User Interface: Strategies for effective human-computer interaction*, Addison-Wesley, 672 pages.
- Storey, M. A., Lintern, R., et al. (2004). *Visualization and Protégé*. 7th International Protégé Conference, Maryland, US.
- Wang, T. D. and Parsia, B. (2006). *CropCircles: Topology sensitive visualization of OWL class hierarchies*. Proc. of the 5th Intl. Semantic Web Conf., Georgia, US, pp.695–708.
- Yee, P., Swearingen, K., et al. (2003). *Faceted Metadata for Image Search and Browsing*. Proc. of the ACM Conf. on Computer-Human Interaction (CHI).

Ontology Management

Semantic Web, Semantic Web Services, and Business  
Applications

Hepp, M.; de Leenheer, P.; De Moor, A.; Sure, Y. (Eds.)

2008, XIX, 295 p. 20 illus., Hardcover

ISBN: 978-0-387-69899-1