Int. J. Human-Computer Studies (2000) **52,** 1111–1133
doi:10.1006/ijhc.1999.0366
Available online at http://www.idealibrary.com on IDEAL®

WonderTools?
A comparative study of ontological engineering tools†

A. J. Duineveld, R. Stoter, M. R. Weiden, B. Kenepa and V. R. Benjamins
Department of Social Science Informatics (SWI), University of Amsterdam, Roetersstraat 15, 1018 WB Amsterdam, The Netherlands. Email: richard@swi.psy.uva.nl

(Received 15 November 1999 and accepted 29 November 1999)

Ontologies are becoming increasingly important in a variety of different fields, such as intelligent searching on the web, knowledge sharing and reuse, knowledge management, etc. Therefore, we expect that the need for tools to support the construction of ontologies will increase significantly in the coming years. In this paper, we investigate several of these tools. We evaluate the tools using two different ontologies: a simple one about university employees, and a second, more complex one, about the structure of a university study. The evaluation was conducted using a framework, which incorporates aspects of ontology buildings and testing, as well as cooperation with other users. Our conclusions are that the usefulness of the tools depends on the level of the users and the stage of development of the ontology.

KEYWORDS: ontology tools.

"Sadly true, but then it's probably still easier to use than Photoshop™ ⋯ We're not generally expecting plumbers and janitors to build ontologies." [Comment of James Rice, principle implementor of Ontolingua].

1. Introduction

Since the beginning of the 1990s, ontologies have become a popular research topic investigated by several Artificial Intelligence research communities, including knowledge engineering, natural-language processing and knowledge representation. More recently, the notion of an ontology is also becoming widespread in fields such as intelligent information integration, information retrieval on the Internet, and knowledge management. The reason for ontologies being so popular is in large part due to what they promise: a shared and common understanding of some domain that can be communicated across people and computers. We expect that the need for ontologies, and therefore for tools supporting the creation of these ontologies, will increase. Because of this increase, several tools will be brought to "market" in the next few years. To give a good overview of the tools already available to users, we evaluated a number of these supporting tools.

†WonderTools website: http://www.swi.psy.uva.nl/wondertools/

1071-5819/00/061111 + 23 $35.00/0

Experience shows that often the bottleneck of building sharable ontologies lies more in the social process than in the technology (Benjamins, 1998). Therefore, tools for collaborative ontology engineering are important.

What are ontologies all about, and what do they intend to support? "Ontologies are content theories about the sorts of objects, and relations between objects that are possible in a specified domain of knowledge. They provide potential terms for describing our knowledge about the domain." (Chandrasekaran, Josephson & Benjamins, 1999). Two parts in this definition are important, namely "content theories of objects and relations" and "domain knowledge". The first is concerned with the whole of objects, let us say objects in a domain, and the relations between them. The second important part is "domain knowledge", and describes all the knowledge you have about some particular topic.

In this paper, we describe the results of a project called: WonderTools,‡ concerned with an analysis of ontology-engineering tools with the aim of assisting users in selecting an appropriate tool for creating ontologies. After an extensive search on the Internet, we considered the following six tools in this project.

- Ontolingua, a web-based tool for making ontologies.
- WebOnto, also a web-based tool, but completely graphical.
- ProtégéWin, a Windows-based tool, also graphical.
- OntoSaurus, a web-based tool, looks much like Ontolingua, but uses the Loom language.
- ODE, a Windows-based tool, a combination of a graphical and text-based tool.
- KADS22, a graphical and text-based tool for building ontologies and reasoning strategies.

After having put a draft version of this paper on-line, we received several comments regarding (new) tools, which we were not aware of at the time of our investigation. Examples are Stanford's JOT editor and the KSL KB merger, the Co4 environment,§ and SRI's General Knowledge-Base Editor.¶

In Section 2, we give background information relevant for this paper. In Section 3, the evaluation framework used will be explained. Then in Section 4, we introduce the different tools we have evaluated. The next section (Section 5) is concerned with the experiment, where we describe the ontologies we tried to build with the tools and how this was done. Finally, we present a discussion (Section 6) followed by the conclusions of our work (Section 7).

2. Background

2.1. ONTOLOGIES

The main motivation behind ontologies is that they allow for sharing and reuse of bodies of knowledge in a computational form. In the knowledge sharing effort (KSE) project

‡WonderTools is an acronym for: Web-based ONtology DEscriptions and Research of its TOOLS.
§http://co4.inrialpes.fr/
¶http://www.ai.sri.com/ ~ gkb

(Neches, 1991), ontologies are put forward as a means to share knowledge bases between various knowledge-based systems. The basic idea was to develop a library of reusable ontologies in a standard formalism, that each system developer was supposed to adopt.

Originally, the term ontology comes from philosophy (it goes as far back as Aristotle's attempt to classify the things in the word) where it is employed to describe the existence of beings in the world. Artificial Intelligence (AI) deals with reasoning about models of the world. Therefore, it is not strange that AI researchers adopted the term ontology to describe what can be (computationally) represented about the world in a program.

Many definitions of ontologies have been put forward in the last decade, but one that characterizes best, in our opinion, the essence of an ontology is based on the related definitions in Gruber (1993) and Borst (1997): "An ontology is a formal, explicit specification of a shared conceptualization". "Conceptualization" refers to an abstract model of phenomena in the world by having identified the relevant concepts of those phenomena. "Explicit" means that the type of concepts used, and the constraints on their use are explicitly defined. For example, in medical domains, the concepts are diseases and symptoms, the relations between them are causal and a constraint is that a disease cannot cause itself. "Formal" refers to the fact that the ontology should be machine readable, which excludes natural language. 'Shared" reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group (Studer, Benjamins & Fensel, 1998).

2.2. CSCW

Computer-supported cooperative work (CSCW) is a hot item today in both research and development. The idea behind it is that people should be able to work together in a group, but do not have to be at one place or time. "CSCW looks at how groups work and seeks to discover how technology (especially computers) can help them work" (Ellis, Gibbs & Rein, 1991). A central point in CSCW is the notion of "GroupWare" (Ellis *et al*., 1991). Because people should interact with each other by some means of a communication model, which is the same for all attendees, software packages have been constructed to help the interaction between the users. The programs and additional hardware are called "GroupWare". Ellis states in his article "The goal of GroupWare is to assist groups in communicating, in collaborating, and in coordinating their activities. [...] We define GroupWare as: computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment."

An example of CSCW we used for our project was the BSCW-server (Basic Support for Collaborate Work). This server, which is located in Germany, supports collaborative work over the web. BSCW provides a "shared workspace", which supports document upload, event notification, group management and much more. To access a workspace, group members only need a standard web browser and a login-name and password.

Because ontology building by a group of people geographically separated over the world is likely to increase in the future, CSCW aspects are becoming more relevant for ontology tools. For this reason, we also looked at CSCW aspects of the tools.

3. Evaluation framework

To evaluate the different ontological engineering tools, we specified a number of relevant criteria. We started out with the system evaluation framework of Kabel (1997), which is intended for the evaluation of software systems. Because we are evaluating a specific type of programs, we added a number of criteria.

We evaluated the tools on three dimensions. First there is a general dimension, which refers to the aspects of the tools that can also be found in other types of programs. This dimension refers to information about the user interface and the different actions the user can perform. For example, the first question of Table 1 evaluates the clarity of the interface. Relevant questions would include: Do you have to explore first or is it immediately clear what options you have and where you can find them? Does the program look organized or do you feel confused (by maybe too much options displayed)? This question is about the first look and feel. Different aspects to consider at question 5 of Table 1 are: When I use an option in a menu, does the program execute the task I expect it to do? Different menus can also use different words for the same thing, for

TABLE 1
Evaluation framework

1. General

1. Evaluate the clarity of the interface
2. Evaluate the consistency of the interface
3. Evaluate the speed of updating after new data is inserted
4. Is there a good overview of the ontology?
5. Is the meaning of the commands clear?
6. Are the changes identifiable by a certain command clear to the user?
7. Evaluate the stability of the tool (crashes, etc.)
8. Does the tool require a local installation?
9. Evaluate the help system

2. Ontology

1. Is it possible to use multiple inheritance?
2. Is it possible to create exhaustive and/or disjoint decomposition? (+ease of doing this)

3.1 Does the tool check new data for consistency with the ontology?

3.2 At what level? (types, disjoint, etc.)

4. Are there example-ontologies available in the tool?
5. Does the tool provide libraries of ontologies that can be reused? Through what operation? (inclusion, union, etc.)
6. Are there high-level primitives?
7. Is there information about the terms used in constructing an ontology in the help system?

3. Cooperation

1. Does the tool allow synchronous editing of the same ontology by different users?
2. Are there ways to lock the ontology?
3. Is it possible to browse an ontology if it is locked?
4. Are the changes made by other users easy to recognize?
5. Is it possible to export the ontology's code in various formats?
6. Is it possible to import an ontology-description from another tool?

instance "class" and "concept". Concerning question 6 of the table, the following questions could be asked: Is it easy to see what changes were made? Do you get feedback? Are the changes made at once or do you have to wait a certain time before your ontology is updated?

The second dimension, the ontology dimension, refers to ontology-related issues found in the tools, such as the amount of help on ontology building and the high-level primitives provided. For example, can you find information on what ontologies are and when it is useful to make one? Are there example ontologies? The last dimension is that of cooperation, which is used to evaluate the tool's support for constructing an ontology by several people at different locations. Some tools provide extra functionality to support this. While with other tools, it is only possible to save the ontology in a certain format and exchange this presentation using email or ftp. To evaluate the tools on these dimensions, we used a checklist, which provides points to keep in mind while we used the tools to construct an ontology. The checklist consists of a number of questions for every dimension. Most questions can be answered by giving a grade for a certain aspect of the tool. The grades used are from 1 to 10, with 1 being extremely bad and 10 being excellent. However, we were more interested in the explanation of why a certain grade was given, than in the grade *per se*, and therefore, in this paper, we used good (+), reasonable (0) and not good (−). Other questions just ask whether a certain option is available, which can be answered by yes or no. Using these evaluations, we will compare certain aspects of different tools to each other.

4. Ontological engineering tools

In this section, we give brief overviews of the six different tools.

4.1. ONTOLINGUA

The Ontolingua system (Rice, Farquhar, Piernot & Gruber, 1997; Farquhar, Fikes & Rice, 1997) was developed in the early 1990s at KSL of Stanford University. The system consists of a server and a representation language. The Ontolingua server|| is located at the university of Stanford; however, we used the mirror stationed in Nijmegen. The Ontolingua server provides a repository of ontologies, allows ontologies to be created and existing ontologies to be modified. The ontologies in the repository can be merged or included in a new ontology. This server is designed to allow several users to cooperate in developing an ontology. Interaction with the server is achieved by using a standard web browser to connect to the server.

The creation of a new ontology is made easier by the possibility to include (parts of) existing ontologies from a repository and the possibility to include primitives from a frame-ontology. This repository consists of a large number of ontologies from different fields. After completion of an ontology, the ontology can be added to the repository for possible reuse.

An important aspect for the Ontolingua system is the ability to design an ontology collaboratively. This allows different users from all over the world to work together in

||http://www-ksl-svc.stanford.edu or at Nijmegen: http://ontolingua.nici.kun.nl

constructing a single ontology. In order to be able to do this, the server uses a notion of users and groups. The owner of an ontology can give certain groups of users read and write access to the ontology. It is also possible to work simultaneously on an ontology in a group session. The server will notify all other users in the same session whenever another user changes something.

The ontologies stored at the server can also be converted to a different format for use in other applications. This allows the users to use the Ontolingua server to create an ontology, export this ontology and then, for instance, use it for a CLIPS-based application. It is also possible to import ontology definitions from a number of languages into the Ontolingua language.

4.2. WEBONTO

WebOnto** (Domingue, 1998) is, as the name suggests, fully accessible via the Internet. It has been developed by the Knowledge Media Institute of the Open University. "WebOnto was designed to support the collaborative browsing, creation and editing of ontologies. In particular, WebOnto was designed to provide a direct manipulation interface displaying ontological expressions using a rich medium. WebOnto was aimed to be easy-to-use, yet have facilities for scaling up to large ontologies. Finally, WebOnto was designed to complement the ontology discussion tool Tadzebao" (Domingue, 1998). WebOnto is a mainly graphically orientated tool for constructing ontologies. "The language used to model the ontologies in WebOnto is OCML. OCML, which stands for Operational Conceptual Modeling Language, was originally developed in the context of the VITAL project to provide operational modeling capabilities for the VITAL workbench." (Motta, 1997). The tool has a number of useful features, like saving diagrams of structures, viewing the relations, classes, rules, etc. separately. Other features include for example, working cooperatively on ontologies, by the use of drawing, and using the broadcast and receive functions. When using these features, different users can make notes and changes in the current overview of the ontology by using various colors. Other users viewing the same ontology will see these changes.

4.3. PROTÉGÉWIN

ProtégéWin†† (Eriksson, Fergerson, Shahar & Musen, 1999) is a Window-based computer program, which should be installed locally. The program is meant for building ontologies of domain models, and has been designed by Stanford's Medical Informatics Section. ProtégéWin has been developed to assist "software developers in creating and maintaining explicit domain models, and in incorporating those models directly into program code. The Protégé methodology, to which the tool belongs, allows system builders to construct software systems from modular components, including (1) reusable frameworks for assembling domain models and (2) reusable domain-independent problem-solving methods that implement procedural strategies for solving tasks" (Eriksson, Shahar, Tu, Puerta & Musen, 1995).

**WebOnto: http://webonto.open.ac.uk
††ProtégéWin: http://smi-web.stanford.edu/projects/prot-nt/

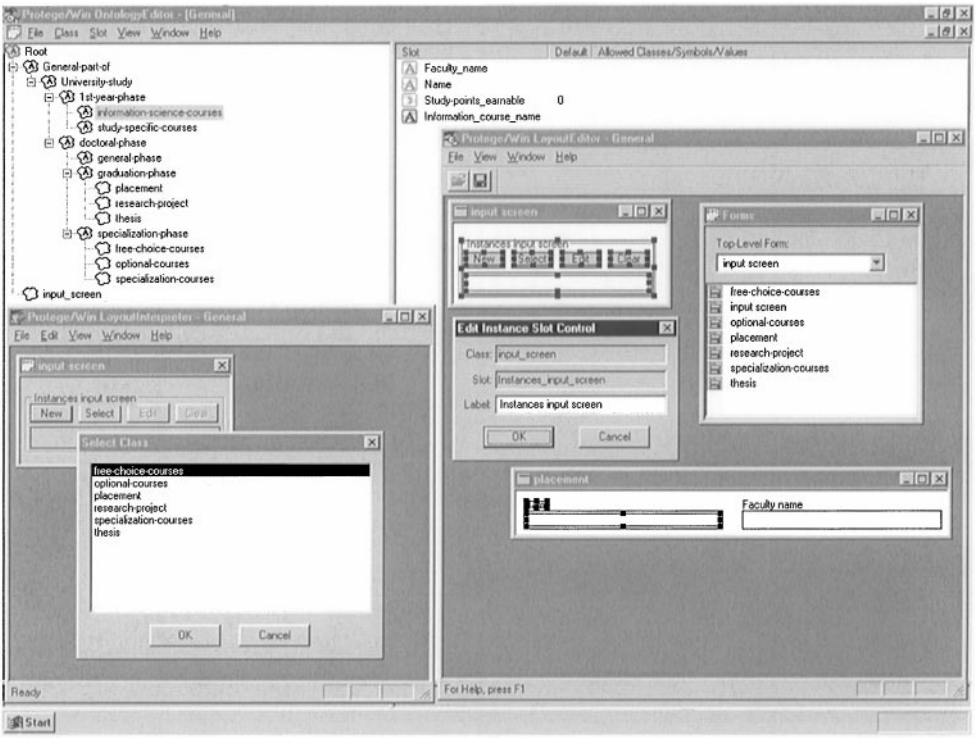

FIGURE 1. ProtégéWin's different parts.

The program consists of three major parts which should be used interchangeably. First there is the "Ontology-editor" which lets you make your own ontology about a domain by just expanding a hierarchical structure, and including abstract or concrete classes and slots.

Based on the ontology built, ProtégéWin is able to generate a knowledge acquisition tool for entering the instances of the ontology (e.g. if the ontology is about university courses, then an instance could be course 215 about knowledge engineering). The KA-tool can be fine-tuned to the needs of a user by using the Layout-editor (see Figure 1).

The last part of the program is the "Layout-Interpreter" which reads the output of the "Layout-Editor" and shows the user an input-screen with a few buttons. These buttons can be used to make the instances for the classes and sub-classes. By clicking the "New" button you can make a new instance. After this you will be asked to select a class to which the instance belongs, and then you can type in this new instance with its characteristics. The whole tool is graphical which is very usable for a naive user.

4.4. ONTOSAURUS

OntoSaurus‡‡ is a web browser for Loom (ISX, 1991) knowledge bases. It provides a graphical hyperlinked interface to Loom knowledge bases. OntoSaurus also provides

‡‡http://sevak.isi.edu:8300/loom/shuttle.html (username and password required)

limited editing facilities for Loom knowledge bases. However, its main function is browsing ontologies. In our project, we evaluated the editing facilities. The underlying knowledge representation language of OntoSaurus is Loom. Loom is a language and environment for constructing intelligent applications. "The heart of Loom is a knowledge representation system that is used to provide deductive support for the declarative portion of the Loom language. Declarative knowledge in Loom consists of definitions, rules, facts and default rules. A deductive engine called a classifier utilizes forward-chaining, semantic unification and object-oriented truth maintenance technologies in order to compile the declarative knowledge into a network designed to efficiently support on-line deductive query processing." Loom is a research project in the Artificial Intelligence research group at the University of Southern California's Information Sciences Institute.

4.5. ODE

Ontology design environment (ODE) (Fernandez, Gomez-Perez, Pazos & Pazos 1999) is an ontology construction tool, which interacts with users (ontology builders) at the *knowledge level* using a set of intermediate representations that are independent of languages used to specify ontologies (LOOM, Ontolingua, F-Logic, etc.). The motivation behind ODE is that humans are much better at formulating ontologies at the knowledge level than by using formal languages. ODE provides ontology builders with a set of tables in which the ontology developer can specify concepts, individuals, relations, functions, axioms, formulas, etc. ODE then generates automatically code for it, currently in Ontolingua and F-logic (Kifer, Lausen & Wu, 1995). ODE forms part of a methodology for the complete life cycle of ontology building, called Methondology (Fernandez *et al*., 1999), developed at LAI of the Technical University of Madrid.

4.6. KADS22

KADS22§§ is a tool for supporting the construction of knowledge models according to the CommonKADS methodology (Schreiber, Wielinga, de Hoog, Akkermans & Van de Velde, 1994). Ontologies form part of such knowledge models (the other part being reasoning models). CommonKADS models are specified in conceptual modelling language (CML). KADS22 is an interactive interface for CML. It provides a graphical interface (much like the familiar Windows programs) with the following functionality: parsing the CML files, pretty-printing, hypertext browsing, generation of the graphical notation, search, glossary generation and HTML generation. KADS22 is under development at SWI of the University of Amsterdam.

5. Experimental set-up and results

This section covers the set-up of our research. As mentioned earlier, in the course of the experiment we have build two ontologies. The first ontology was a very simple one about the types of persons belonging to the academic world, such as students, Ph.D. students,

§§http://swi.psy.uva.nl/projects/KADS22/index.html

undergraduate students, assistant-, associate- and full professors, etc. These concepts were modelled using the sub-class-of relation as illustrated in Figure 2 in OntoSaurus. The aim of this exercise was to get acquainted with the various tools and to evaluate the amount of foreknowledge needed to use the tools (see the Discussion section). At the beginning of this project, we were not familiar with these tools for building ontologies, but we were familiar with computer-supported cooperative work (CSCW), human–computer interaction, evaluation methods and literature on ontologies. Based on our experience (including the problems encountered), we made a brief evaluation report and sent that to the developers of the tools. The reason was to get additional explanation on the use of the tools and to prevent us from carrying on misinterpretations of the tools to the second phase of our project (e.g. in some cases we thought that multiple inheritance was not possible, while it actually was possible). After this first stage, we dropped the KADS22 tool for inclusion in the more thorough evaluation study because we had practical problems in using it conveniently for the simple ontology.

5.1. THE SWI ONTOLOGY

The second ontology concerned the area of "university studies in the Netherlands". Our aim has been to build a general ontology of this domain, which would then be instantiated for the specific university study SWI (Social Science Informatics). The populated ontology can serve as an on-line information system for providing potential and actual SWI students with information about the structure and content of the study. On the other hand, the ontology of Dutch university studies, can be used for modelling any other Dutch study, such as for example Computer Science, Spanish Literature, or Psychology. The ontology could be used by department coordinators, who would have to fill the ontology with their study-specific courses. In turn, the populated ontology, along with one of the tools discussed here in browse-mode, might then serve as an information service to interested students.

The general ontology consists of two main structures: a part-of hierarchy, modelling the structure of the study, and various taxonomic hierarchies modelling the different types distinguished in the domain. Examples are shown in Ontolingua for various taxonomic links (Figure 3), and in WebOnto for the types of courses (right part of Figure 4). The taxonomy of courses is domain independent until a certain level. After the doctoral course level (Figure 4), the structure becomes domain-specific, because not all university studies have a distinction between specialization courses and general courses. The top-most class of this taxonomic structure is the `course` class. Since all courses have some form of teaching and some form of examination, these classes (e.g. `course_form` and `examination`) provide the values for the slots `has_form` and `has_examination`. Both the classes `examination` and `course-form` have several instances. The general part-of structure has `university_study` as the root of the structure (see Figure 1 and left part of Figure 4). Each university study consists of a `first_year_phase` and a `doctoral_phase` (note that this is the structure in at the universities in The Netherlands). The doctoral phase then again is decomposed into a `specialization` and a `graduation_phase`. The graduation phase in its turn has the following parts: `thesis`, `placement` and `research-project`. Each of the different phases has a slot `has_course`. The values of these slots are provided by the corresponding courses

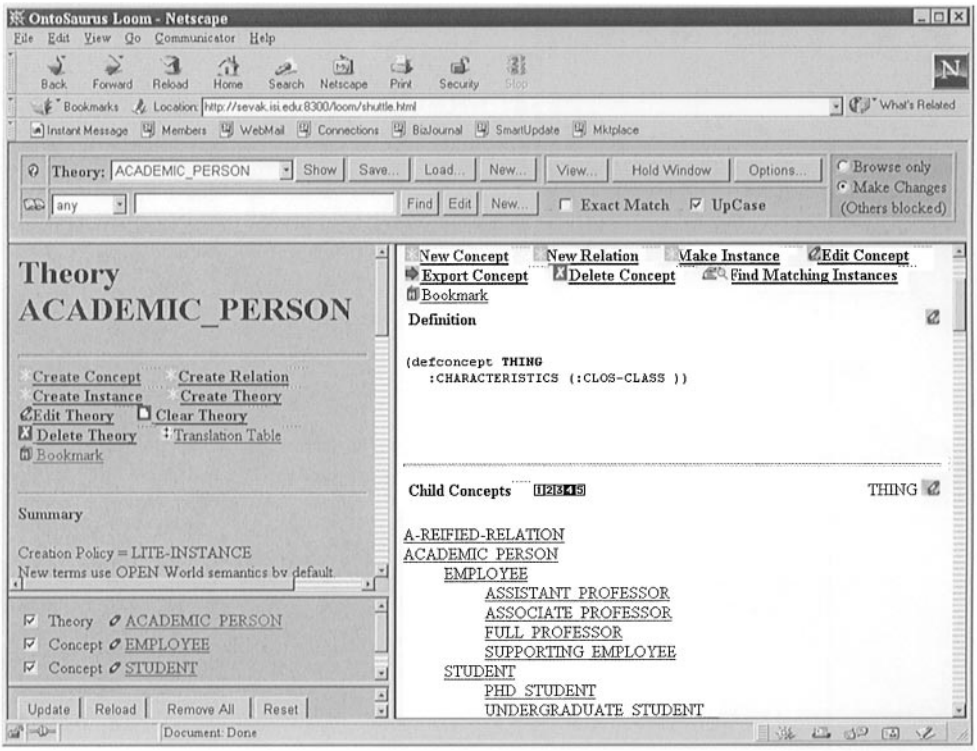

FIGURE 2. Person belonging to the academic world as shown in OntoSaurus.

(instances) of the earlier mentioned taxonomic hierarchy. Eventually, this general part-of structure can be used to model curricula of different studies, by selecting from courses already modelled in the course taxonomic hierarchy.

We tried to create the ontology with five of the earlier mentioned tools. During our work with the tools, we have contacted most of the creators or administrators of the tools, to correspond with them about the use of the tools and the underlying languages, and to ask for help in case of problems. While building the ontologies, we evaluated the various tools according to the evaluation framework, as presented in Section 3. The framework includes questions about the interface, ontologies, and the support for cooperation provided by the tools. In the rest of this section, we present the results of the evaluation for each of the tools separately, and in the next section, a comparison will be made between the tools.

5.2. EVALUATION OF ONTOLINGUA

5.2.1. General

Ontolingua (Figure 3) can be accessed using a standard web browser. The Ontolingua interface is then presented in the browser's window. Ontolingua's interface consists of two frames. The top frame has a number of icons and menus. The bottom frame is used to

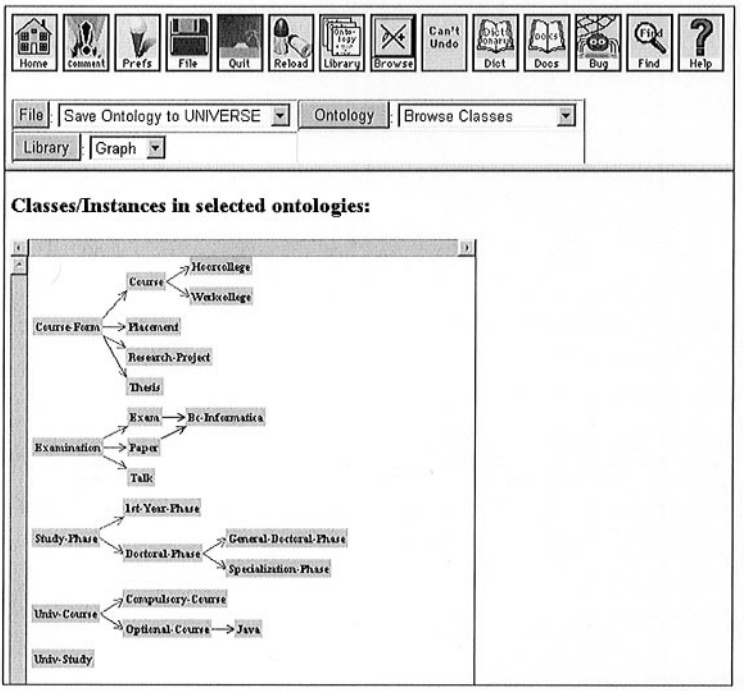

FIGURE 3. Ontolingua's ontology grapher, showing the SWI-ontology.

show information and to input new data. This interface is different from the standard used by Windows, but it is easy to understand and use. The meaning of the comments is clear and in case one wants to have more information about a certain command, there is a help facility available. There is also a guided tour to get familiar with the basic commands. The bottom frame is updated, whenever new data is inserted or something else is changed. This ensures that the information presented to the user is always up to date. If the Ontolingua server is used by several users to work on the same ontology, changes invoked by another user are updated as soon as anything is clicked. Ontolingua does not use the standard graphical overview of the ontology (rectangles to represent classes, lines to represent relations, etc.). Instead there is a graphical browser. This browser can be used to get an overview of the taxonomy of the concepts in the ontology along with the instances. It is not possible to get a graphical overview of other aspects of the ontology (other relations).

5.2.2. Ontology

In Ontolingua, it is easy to use the principle of multiple inheritance. Slots in an instance are inherited from multiple parents, like they should be. It is also possible to define compositions as exhaustive and/or disjoint. The Ontolingua tool comes with a repository of ontologies. These can be merged or otherwise included into a new ontology. These ontologies can also be browsed to find out how to implement certain features of an

ontology. Also, there are a large number of primitives available in the frame ontology. These primitives can be imported into the ontology to represent certain types of relations. Using these primitives, it is possible to define a decomposition as for instance an exhaustive one. However, it is not always clear how one should use these primitives. There is only a short description about the primitives and no help on how to implement their use in an ontology.

There is no specific help on ontology building, apart from the basics given in the guided tour. This is certainly an aspect of Ontolingua that could be improved. Even for people acquainted with ontology building, it can be hard to find out how to implement a certain feature in Ontolingua.

5.2.3. Cooperation

Since Ontolingua is intended to be a tool for cooperative ontology construction, there are many features to support this. It is possible for several users to edit the same ontology simultaneously. The changes made by someone will be visible to all other users, after they click on anything. This makes sure that the ontology a user sees on his screen is consistent with the ontology on the server, even if something has changed as a result of an action of a different user. In addition, all users in the same shared session will be notified of the changes made by another user. There is a shared undo-list available. This means that any user can undo changes made by all other users. There is also a redo-list available.

It is possible to define the types of access other users have to an ontology. This means one can lock an ontology for editing, while it is still available for browsing to other users. The Ontolingua server comes with excellent importing and exporting capabilities. This can be done in a number of formats.

5.3. EVALUATION OF WEBONTO

5.3.1. General

When viewing WebOnto (Figure 4) in the light of the evaluation framework, a few characteristics stand out. First on the general dimension, the interface is rather clear. The interface is made consistent with standard win95/98 interfaces. Therefore, the commands are clear to use. WebOnto uses besides the standard file/edit menu, more specific menus like an ontology menu, operations menu, annotation menu and broadcast menu. The ontology menu consists of ontology-specific operations like selection of ontologies and viewing classes, instances etc. The operations menu is used to make inferences about the ontology through actions like tell, ask and evaluate. The annotation and broadcast menus finally are used to work collaboratively on an ontology. The structure of these menus is very clear and easy to use. When one is editing an ontology, it will be locked for other users to edit. This option is provided by the use of a browse and edit mode, selectable in the edit-menu. During an edit session the ontology is updated, when something is modified within the ontology. The overview of the ontology presented in WebOnto is one of the better overviews provided in relation to the other tools, not only can the entire ontology be viewed, but also selections of the ontology, for example only the classes, or only the instances, etc. All these structures are presented in graphical tree structures. Especially when building large ontologies, the structures can become rather confusing; however due to the ability to view only parts of it, the user is able to get a clear

view of the domain. Furthermore, the system is rather stable, no serious problems when working with the tool, were encountered.

5.3.2. Ontology

In WebOnto the user can create structures including classes with multiple inheritance. This can be done graphically. All the slots are inherited in the correct way. It is also possible to construct disjoint, exhaustive compositions; however for now the only way of doing it, is by stating it in the underlying knowledge representation language. The tool does check newly entered data by checking the integrity of the OCML code. In WebOnto there are many other ontologies, which can be browsed freely, all these ontologies import a base-ontology. When viewing this base-ontology, all underlying classes are presented as well, this way ontologies can be reused. The base-ontology also includes numerous primitives, like sub-class-of, logical operators, integers, etc. These primitives can be used to define for example sets and sub-classes. However a predefined part-of relation is not included in this base-ontology.

5.3.3. Cooperation

WebOnto comes with a feature called broadcast and receive mode. When one enables the broadcast mode, and then enters the edit mode, other users in receive mode can view the

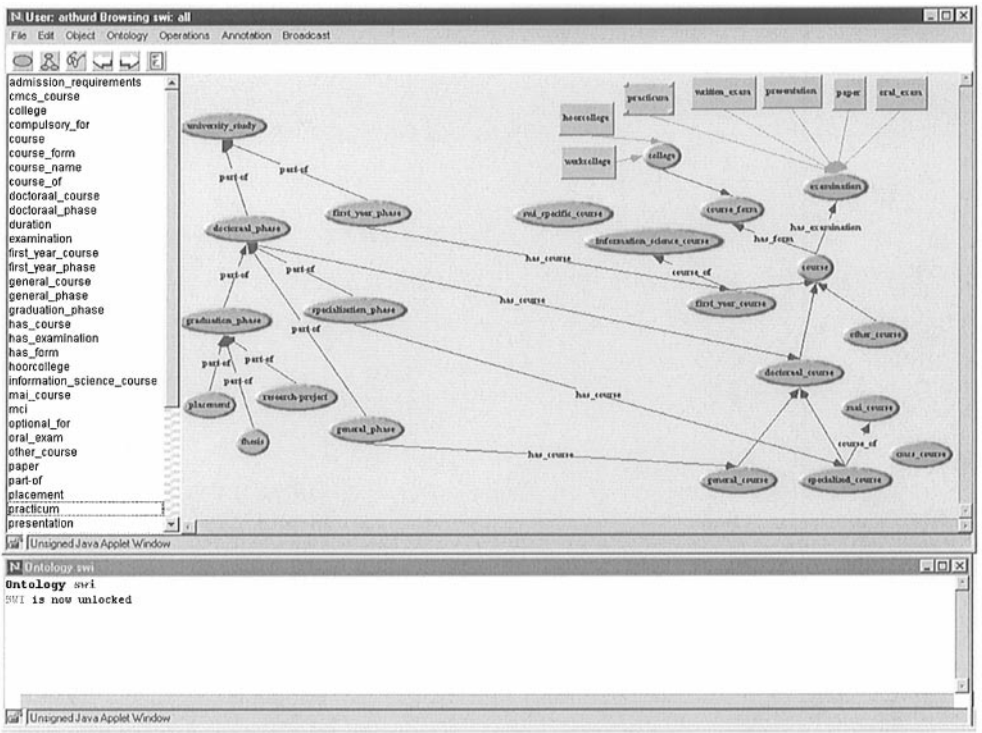

FIGURE 4. Graphical structure of SWI ontology as presented in WebOnto. Unlabelled arcs are taxonomic links.

changes the broadcasting party is making. However, only one user at a time can edit an ontology, when a user is editing, the ontology becomes locked. In the locked state, the ontology can still be browsed through by other users. Unfortunately, there is no direct way of exporting ontologies made in WebOnto, the only possible way up till now is viewing the OCML source code, and then copy/paste it to another program. However, in working with the tool, we noticed it is still under construction, and it was even adapted to our comments, when we encountered problems. WebOnto also includes an annotation tool, which consists of a "drawing pencil". Ontology builders can use it to draw anything (arrows, circles, text, etc.) in the graphical window. Other users who are at that time in receive mode, will see these annotations in real time.

5.4. EVALUATION OF PROTÉGÉWIN

5.4.1. General

ProtégéWin (Figure 1) has a nice graphical interface which should be easy to use for all users. The ontology structure is made up just like a hierarchical directory structure. Classes, sub-classes and slots can be added and edited by using the standard right-mouse click.

When we look at the consistency of the program, we see that this is comparable to the clarity of the interface. On each class (sub-class or super-class) you can use the same options as mentioned above. No inconsistency was found when we used this tool. Due to the segregation of the parts (Ontology-editor, Layout-editor or Layout-interpreter) some confusion arose about the functionality of the parts.

The ontology was updated in real time in the Ontology-editor. The updating of the other parts of the complete program was conducted by saving the ontology in the Ontology-editor and then opening it in the Layout-editor.

As mentioned in the first paragraph above, the ontology is shown in a hierarchical structure, what gives a good overview of the ontology. Inherited slots are easy to recognize in the structure by code colouring.

The commands used in the tool are not always clear to all users. When you are new to ontology building you probably have to use the help facility very often, but even when you are quite experienced in using and creating ontologies you will find some things that are not clear. In the help facility there are a lot of explanations. Experienced users should have enough information in the help facility.

This tool was very stable in comparison to some other tools ProtégéWin is not very comparable to the web-based tools, because they are dependent on the traffic on the Internet, while ProtégéWin is a locally installed tool.

5.4.2. Ontology

Multiple inheritance is possible, but might lead to problems if classes inherit slots with identical names. The tool checks real time in the Ontology-editor if there is still consistency when giving new data. In the other parts of the tool there are no consistency checks because they are only concerned with the layout of the input-screen, and not with the ontology itself.

An overview of the different aspects of the ontology is available in ProtégéWin, in the main screen of the Ontology-editor. There is an example ontology available in the help

facility in HTML format. In the help facility some important terms concerning ontologies are explained, but not enough to give a naive user a good overview of the ontology field. The example is very clear and gives a good overview of what you can do with the program. A library of ontologies for reuse is, however, not available. You could use the example for parts, but this is not ideal.

5.4.3. Cooperation

ProtégeWin is a tool, which should be installed locally on your computer, as mentioned earlier. That is why it is not a real CSCW tool. Synchronous collaboration is not possible with this tool. The only exchange of information can be done by saving your ontology and reading it into another copy of ProtégéWin. Because of this, it is not possible to work on an ontology together at the same time. Ontology locking and browsing during locking are not applicable to this tool.

Changes to the tool are not easily recognizable to another user, unless you have an old copy of the ontology on your computer. This feature of cooperation will be available in the next version of Protégé, which will be a Java program, called Protégé2000. The three different parts of the program will be merged into one tool in this next version. In this version of ProtégéWin, it is not possible to export or import to/from other types of format. This is not handy if you want to use ontologies made in one of the other evaluated tools.

5.5. EVALUATION OF ONTOSAURUS

5.5.1. General

The OntoSaurus Browser (Figure 2) is an ontology environment based on the higher level programming language "Loom". When you work with OntoSaurus, you will find a small description of how the environment works. A few examples of ontologies are given and it is possible to browse through them. How the examples were made is not shown however. With OntoSaurus it is very easy to browse through ontologies, even if they are very complex. The browser has four frames. In one frame you can make special bookmarks of the ontology, which can make browsing easier and faster. In the top frame you can choose which ontologies, called theories, you want to see. On the right is a frame that shows the main concepts of an ontology. A concept (or class) is shown as a hyperlink and by clicking you will see underlying concepts or other details. The OntoSaurus Browser has two modes. One mode for browsing through an ontology and one for making changes. If you wish to make changes, you select that mode and you will block out others trying to edit in the same environment. Different editing options will become available. At first a naive user may not find it easy to edit. One must fill in the right edit boxes with the right text. For example: If you want to make a subconcept, you will have to fill in the parent in the right concept box. Minor changes, like changing details of a concept, are easy to make in Ontosaurus. For building an ontology, complex ontologies in particular, it is necessary to understand the Loom language. Most users build the ontology in Loom in a different editor and then import it into OntoSaurus and for browsing and correcting. One can use OntoSaurus on the web with either a locally installed version of Loom or the Loom version installed at the ISI site.

5.5.2. Ontology

Because OntoSaurus uses the Loom language, it has all the power the Loom language has to offer, like the Loom classifier. Automatic consistency checking and deductive reasoning support and several other facilities are also provided. Things like multiple inheritance, exhaustivity and constraints can be easily expressed in the Loom language. However, in OntoSaurus one has to know how to fill in the right edit boxes. There is no help about how to build an ontology. Thus, for naive users it might not be easy to build ontologies. However, experienced users can use OntoSaurus to swiftly look through or edit details. To get an overview of an ontology, one must click on the parent (higher level concept) of the theory. The overview consists of hyperlinked concepts. Some may find a graphical overview easier to work with, although complex theories might be more difficult to overlook in a purely graphical way.

5.5.3. Cooperation

It is possible to work synchronously on an ontology. When you make changes you block other users. If there is an edit lock, you can see which user is editing and at what time the lock was created. Two options are suggested: You can send an email to the other person, or you can override the lock. This may be necessary if the other person is not on-line at that moment. After making changes, you must update your ontology and go to browse mode. Then another person can use the updated ontology. In working with OntoSaurus, we did not find a way to recognize the changes that were made and by whom the changes were made. OntoSaurus supports further cooperative work, because it is possible to import and export ontologies in other languages, such as Ontolingua, KIF, Stella and IDL. At the time we were using OntoSaurus, the tool only read Loom, but you could save your ontology in other languages.

5.6. EVALUATION OF ODE

Ontological Design Environment (Figure 5) is still under development and was at the time of our evaluation, not publicly available. We decided to include ODE nevertheless

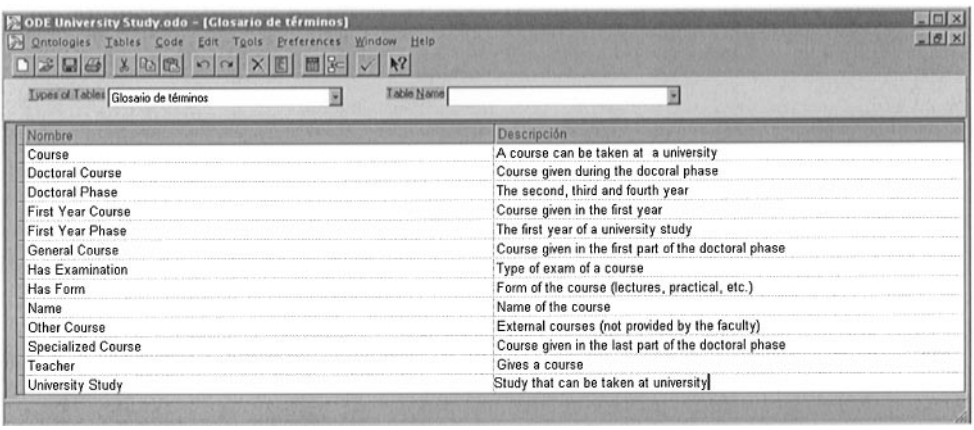

FIGURE 5. Glossary of terms used in our ontology shown in ODE.

because it promises to be a useful tool for non-power users. The results reported in this section are based on building the ontology in a tutorial session with one of the developers.

5.6.1. General

In ODE, an ontology builder does not manipulate a knowledge representation language. Instead, the ontology is built at the conceptual level and later automatically translated into a target language. The interface of ODE consists of a set of tables, such as tables for a glossary of terms, a data dictionary, concepts, relations, axioms, etc. The use of tables is consistent, although not always the most intuitive way to express knowledge (e.g. to specify a class hierarchy of concepts in a table). In the current version of ODE no graphical interfaces are available, but these are currently being developed. An advantage of the use of tables is that the user does not need to have any knowledge about underlying knowledge representation languages (such as Ontolingua, LOOM, OCML, etc.). In fact, experience of the ODE developers is that domain experts are able to evaluate the tables (i.e. the ontology), and detect errors, which is unlikely when using directly knowledge representation languages.

While completing the tables, much support could be gained from providing the user with alternatives to choose from. For instance, while filling a table of relations between concepts, rather than typing the concepts each time, selecting them from a menu automatically generated from the glossary of terms, would be a much more comfortable way of working. ODE's current version does not yet support this feature, but it is planned before it will be publicly available.

In its current version, it is rather difficult to get a clear overview of what has been modelled so far. The planned addition of graphical tools should bring significant improvements.

ODE has the possibility to define explicitly a conceptual model of the ontology itself. Based on this conceptual model (actually, a meta-model of the ontology), the needed tables are automatically constructed. In many situations, a specific set of standard tables is probably sufficient to build an ontology. However, if specific needs arise, the set of tables can be changed by changing the conceptual model. ODE can thus be configured for specific purposes.

5.6.2. Ontology

ODE allows multiple inheritance (a concept may have more than one parent concept). However, ODE does not include an inference engine and thus does not perform the inheritance itself. The reason is that ODE uses translators to generate code for the ontology in a knowledge representation language, and often these languages effectuate the multiple inheritance (e.g. Ontolingua).

ODE includes tools for consistency checking and verification of the ontology. For instance, it checks whether instances of disjoint classes are actually disjoint (i.e. an instance cannot belong to two disjoint classes at the same time). It also checks whether terms used in the ontology have been properly defined (using the data dictionary).

High-level ontological primitives include sub-classes-of, disjointness, exhaustivity and partition. The part-of relation is not a primitive with associated semantics, but of course, it can be defined by the user. Ontology-specific help is available as well as some example ontologies.

5.6.3. Cooperation

ODE needs to be installed locally on a computer (currently a PC with Windows95, 98, NT). Therefore only asynchronous cooperation is supported. Change management between different versions of the ontology is not supported. Users have to keep track themselves of what changes have been made since the last version. Internally, the ODE ontology is stored in a relational database, which facilitates its integration with traditional software. Concerning export and import facilities, the current version of ODE generates Ontolingua and F-logic and is also able to import ontologies in these languages.

6. Discussion

As in many software development projects, also in the tools we have evaluated, documentation has no high priority during the development phase. In general, this makes the use of the tools troublesome for naïve, inexperienced users. The lack of full documentation convinced us to contact the developers of the various tools several times, even for trivial questions.

The goal of building the first ontology (about university persons) was to get acquainted with the tools and to assess the amount of foreknowledge needed. Based on the results, we have now a global idea of how difficult it is to learn to work with the tools and about the amount of knowledge required of the underlying knowledge representation language. The results are summarized in Figure 6.

ProtégéWin and ODE do not require much knowledge of the underlying representation language, and are therefore aimed at non-power users. ProtégéWin also is easy to learn as its interface is straightforward. ODE is more difficult to learn because it uses a table format. The price that ProtégéWin pays is that is provides less high-level primitives, and does not allow one to model axioms. Ontolingua, OntoSaurus and WebOnto all use the power of an underlying knowledge representation language. This allows for more complex modelling, but the price is that users need to know these languages before they can build ontologies (this is especially the case with OntoSaurus, which uses Loom. But then, OntoSaurus was mainly developed as a browsing tool).

In the rest of this discussion, we will compare the different tools according to the three dimensions of the evaluation framework: general (interface), ontologies, and cooperation.

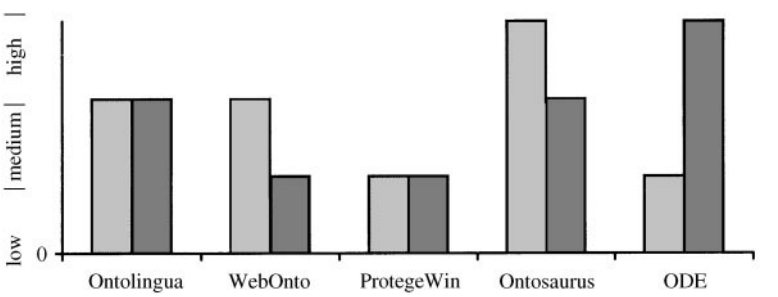

FIGURE 6. Difficulty of learning the tools. ☐ Foreknowledge needed of underlying knowledge representation language; ■ difficulty of learning

TABLE 2

A summary of the results. A plus (+) means positive, a zero (0) means reasonable, a minus (−) is negative, "NA" stands for not applicable and a questionmark means that we have been unable to find out

Criterion	Ontolingua	WebOnto	ProtégéWin	OntoSaurus	ODE
1. General					
1.1 Interface clarity	−	+	+	−	−
1.2 Interface consistency	+	+	+	+	+
1.3 Speed of updating	−	0	+	−	+
1.4 Overview	0	+	+	+	−
1.5 Meaning of commands	+	+	+	+	0
1.6 Identification of changes	0	0	0	0	0
1.7 Stability	+	+	+	+	−
1.8 Local installation	No	No	Yes	Yes/no	Yes
1.9 Help system	+	−	+	+	−
2. Ontology					
2.1 Multiple inheritance	Yes	Yes	Yes	Yes	Yes
2.2 Decomposition types	+	+	−	+	+
2.3.1 Consistency checking	+	+	+	+	+
2.3.2 Level of checking	?	0	0	?	+
2.4 Example ontologies	+	+	0	+	0
2.5 Reusable ontologies	+	+	−	+	−
2.6 High-level primitives	+	+	−	+	+
2.7 Ontological help	−	−	−	−	+
3. *Cooperation*					
3.1 Synchronous editing	+	+	−	+	−
3.2 Ontology locking	+	+	−	+	−
3.3 Browsing when locked	+	+	NA	+	NA
3.4 Change recognition	−	−	−	−	−
3.5 Export facilities	+	−	−	+	0
3.6 Import facilities	+	−	−	+	+

Table 2 gives a summary of the results, in which we compiled the 0–10 scale into a 3-level scale of (+, 0, −). A plus (+) means positive, e.g. the feature/characteristic is available or properly implemented. A zero (0) means reasonable, e.g. the feature is available, but it is difficult to use. A minus (−) is negative, e.g. the feature is not supported or not correctly implemented. "NA" means not applicable and a questionmark means that we have been unable to find out.

6.1. GENERAL

Tools that interact with the user through web browsers generally have an easy-to-learn interface. A complicating factor is that often so many different options are available that one gets lost easily. This is also related to the fact that the underlying knowledge representation languages provide many possibilities, each of which needs to be reflected in the user interface. There is definitely a trade-off between the number of possible

options for the user and the difficulty of using the system. Most systems provide an interface to implement the most used options and allow the user to add more complicated options by writing directly in the systems language/code.

For getting an overview of the ontology, different manners are used. Ontolingua, OntoSaurus and ProtégéWin provide indented lists for representing the class hierarchy. Ontolingua allows in addition also to show it graphically in a tree of class names. WebOnto provides a graphical overview with classes represented as ellipses, instances as rectangles and relations as (labelled) arcs. Graphical interfaces are more insightful for small- and medium-sized ontologies. However, for large ontologies fully graphical interfaces tend to become confusing (because the screen gets filled by numerous lines, ellipses and rectangles), and straightforward indented lists are better candidates. On the other hand, this means it is more difficult to see relations other than taxonomic relation between classes. Several tools provide the possibility to graph specific parts of the ontology (e.g. starting from a particular sub-class).

ProtégéWin and ODE are programs which should be installed locally and both work under the Windows operating system. Because of this, collaborative work is only possible by exchanging ontologies through email, ftp or the like. On the other hand, the tools that work locally are much faster. Especially ProtégéWin, which is very stable, has a nice interface and can be used for the whole trajectory of building ontologies and is also usable by quite naive users. The ODE versions we have used were all very buggy and could not be used for extensive evaluation. The system should also be usable for naive users, but there was not enough information available to check this. Of the web-based tools, WebOnto was the fastest in updating changes made to the ontology. OntoSaurus can be used both with a locally installed version of Loom (for faster updating) or with the version installed at the ISI site.

Another noteworthy aspect of the tools is the overall poor help systems. Although user guides are available, the tools should have properly functioning help within the system. Because the field of ontological engineering tools is rather new, extensive help would appear to be a key element of the tools. In WebOnto and ODE however, there is no general help about the tool available. ODE, although in Spanish, is the only tool providing some help about ontologies. In general however, the help facilities or documentation are in most cases far from optimal. As mentioned earlier, this is mainly due to the fact that academic projects usually do not have budget for extensive documentation.

Another way of supporting users in creating is through examples. Web-based tools provide easy access to existing ontologies. Locally installed tools have to include example ontologies in their release.

6.2. ONTOLOGIES

All the evaluated tools supported the use of multiple inheritance correctly. The easiness of creating an exhausitve and/or disjoint decomposition differed for all tools. We did not find out how to implement this in ProtégéWin. But all other tools supported this in some way.

Concerning error checking, when the user enters new data, this data can be checked for syntax errors, but also for more semantic errors like disjointness of classes. Most tools

support checking of syntax errors, but few tools check for semantic errors. ODE is claimed to do that as well as OntoSaurus and Ontolingua. Ontolingua comes with a large repository of ontologies. Also, a large number of primitive are available in the Frame ontology. This allows a user to import definitions and parts of other ontologies while building a new ontology. However, it is not always clear how the primitives from the Frame ontology should be used. More help on this aspect is definitely needed.

Concerning high-level primitives, the well-known sub-class-of relation is supported by all tools, and several of them also provide for disjoint and exhaustive class decompositions (Ontolingua, OntoSaurus, WebOnto, ODE). High-level primitives are usually defined in a *base ontology*. Any new ontology imports this base ontology and the primitives can be reused when creating a new ontology. Ontolingua imports the Frame Ontology, OntoSaurus imports Loom and WebOnto imports the OCML base ontology. None of the tools provides the "part-of" relations as a primitive, that is, the part-of relation is not defined in the various base ontologies. Having asked the creators, this is due to the non-consensus on the semantics of the part-of relation (studied in the field of mereology).

All tools, except ProtégéWin and ODE, include libraries of sample ontologies that can be browsed and reused in new ontologies.

6.3. COOPERATION

Ontolingua is the only tool that supports full synchronous editing of ontologies. This is a feature that is important if one wants to construct an ontology collaboratively. Ontolingua also has other useful features that support collaborative design, like the shared undo-list. Another example is the good import and export capabilities of the tool. In addition, all users in a shared session will be notified whenever a user changes something in the ontology. We found Ontolingua to be one of the best tools for collaboratively working on ontologies in the last phase of its development. Other web-based tools support synchronous editing, but use locking to prevent overwriting other users' changes. WebOnto additionally supports drawing annotations onto the ontology, which works in a similar way as Netmeeting. Locked ontologies can still be browsed by other users.

Changes made by other users cannot be identified in any tool. This is a major disadvantage, because this would enhance cooperation by showing the different users' changes. The creators of Ontolingua state that this feature is possible, but we did not encounter it while using the tool. We think this may be caused by us using the Nijmegen mirror instead of the more frequently updated Stanford site. For future versions of the tools we suggest using colour coding for indicating the changes made.

Importing and exporting different ontologies is a useful feature when working on an ontology with different tools (e.g. for different stages in the development of the ontology). Recently, a new standard for ontological engineering tools has emerged: the Open Knowledge Base Connectivity (Chaudhri, Farquhar, Fiker, Karp & Rice, 1998). OKBC is a protocol through which OKBC compliant tools can interchange ontologies, making exporting and importing ontologies easier. In this paper, we did not investigate the different types of ontology integration the tools support (e.g. composition, inclusion, union, etc.).

7. Conclusion

Ontologies are becoming more and more important in many different fields. Because of this increase, the need for tools supporting the creation and maintenance of ontologies will increase as well, both for naive and experienced users. We have evaluated five tools to get a good insight into how they work and for what users they are suitable. We also looked at which tools were best suited for the various parts of the process of creating an ontology.

The ontological engineering tools can be classified into different types of tools: locally installable tools and tools accessible through the Internet. Tools, which are installed locally, do not support synchronous collaboration for obvious reasons. Web-based tools however do support synchronous editing and/or discussing of ontologies.

The tools we have evaluated were not all equally suitable for the complete trajectory of building and maintaining ontologies. We found that WebOnto, ProtégéWin and ODE were best suited for the conceptualization and formalization phase in ontology development. These tools are also better suited for the less experienced users. Ontolingua and OntoSaurus appear to be better suited for later phases where only relatively small revisions and modifications are required. Due to the more powerful underlying knowledge representation languages, these tools are not, as James Rice put it, very appropriate for "plumbers and janitors" (i.e. inexperienced users).

Because the different tools have all their pros and cons, it is impossible to select an ideal tool at the moment. Our conclusion is that for less experienced users WebOnto and ProtégéWin are better suited, because they require little knowledge of the underlying knowledge representation language, and are easy to learn. These two dimensions are important for beginning ontological engineers. However, when users have gained more insight with regard to the underlying KR languages, Ontolingua and OntoSaurus might give the users better support in creating complex ontologies.

Our final conclusion is that current ontology tools are not yet ready for direct use by domain experts, but neither do they require knowledge representation experts. In other words, we are halfway. We also noticed that CSCW features more and more find their way into these tools, anticipating geographically distributed development teams.

We would like to thank the following people for their contribution: James Rice, John Domingue, Enrico Motta, Ray Fergerson, Eduard Hoenkamp, Thomas Russ, Asunción Gómez Pérez, Oscar Corcho, Anjo Anjewierden and Bob Wielinga.

References

BENJAMINS, V. R. (1998). The ontological engineering initiative $(KA)^2$. In *Formal Ontology in Information Systems*, GUARINO, N. (Ed.) pp. 287–301. Amsterdam: IOS Press.

BORST, W. N. (1997). Construction of Engineering Ontologie, PhD Thesis, University of Twente.

CHANDRASEKARAN, B., JOSEPHSON, J. R. & BENJAMINS, V. R. (1999). What are ontologies, and why do we need them. *IEEE Intelligent Systems*, 20–26.

CHAUDRI, V. K., FARQUHAR, A., FIKES, R., KARP, P. D. & RICE, J. P. (1998). *Open Knowledge Base Connectivity* 2.0. Knowledge Systems Laboratory.

DOMINGUE, J. (1998). Tadzebao and WebOnto: discussing, browsing, and editing ontologies on the web. *In Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management, KAW'98.* Banff, Canada.

ELLIS, C. A., GIBBS, S. J. & REIN, G. L. (1991). Groupware, some issues and experiences. *Communications of the ACM*, **34**.

ERIKSSON, H., FERGERSON, R., SHAHAR, Y. & MUSEN, M. A. (1999). Automatic generation of ontology editors. *Proceedings of KAW-99* (to appear).

ERIKSSON, H., SHAHAR, Y., TU, S. W., PUERTA, A. R. & MUSEN, M. A. (1995). Task modeling with reusuable problem-solving methods. *Artificial Intelligence*, **79,** 293–326.

FARQUHAR, A. & FIKES, R. & RICE, J. (1997). The Ontolingua Server: tool for collaborative ontology construction. *International Journal of Human-Computer Studies*, **46,** 707–728.

FERNÁNDEZ, M., GÓMEZ-PÉREZ. A., PAZOS, J. & PAZOS, A. (1999). Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems*, 37–46.

GRUBER, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition Journal*, **5**, 199–220.

ISX Corporation (1991). *LOOM Users Guide*, Version 1.4.

KABEL, S. C. (1997). *System evaluation framework*. M.S. Thesis, University of Amsterdam.

KIFER, M., LAUSEN, G. & WU, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*.

MOTTA, E. (1997). *Reusable components for knowledge modelling*. Ph.D. Thesis, The Open University.

NECHES, R., FIKES, R. E., FININ, T., GRUBER, T. R., SENATOR, T. & SWARTOUT, W. R. (1991). Enabling technology for knowledge sharing. *AI Magazine* **12,** 36–56.

RICE, J., FARQUHAR, A., PIERNOT, P. & GRUBER, T. (1996). Lessons learned using the web as an application interface. In *CHI96*.

ROBINSON, M. (1991). Computer supported cooperative work: cases and concepts. *Proceedings of Groupware '91, SERC*.

SCHREIBER, A. Th., WIELINGA, B. J., de HOOG, R., AKKERMANS, J. M. & VAN DE VELDE, W. (1994). CommonKADS: a comprehensive methodology for KBS development. *IEEE Expert* **9**, 28–37.

STUDER, R., BENJAMINS, V. R. & FENSEL, D. (1998). Knowledge engineering, principles and methods. *Data and Knowledge Engineeriong*, **25**, 161–197.

SUCHMAN, L. (1989). *Notes on computer supports for cooperative work*. Working Paper WP-12, Deptartment of Computer Science, University of Jyvaskyla, SF-40100, Jyvaskyla, Finland.

Paper accepted for publication by Editor, Dr B. Gaines