Escórcio, L. and Cardoso, J. "Editing Tools for Ontology Construction", in "Semantic Web Services: Theory, Tools and Applications", Idea Group. (scheduled for March 2007).

Editing Tools for Ontology Construction

Jorge Cardoso
Department of Mathematics and Engineering
University of Madeira
9000-039 Funchal, Portugal
jcardoso@uma.pt

Ana Lisete Nunes Escórcio
Department of Exact Science and Technology
Escola Básica e Secundária do Carmo
9300 Câmara de Lobos, Portugal
li7@netmadeira.com

Abstract: This chapter gives an overview of some editing tools for ontology construction. At the present time, the development of a project like the one of building an ontology demands the use of a software tool. Therefore, it is given a synopsis of the tools that the authors consider more relevant. This way, if you are starting out on an ontology project, the first reaction is to find a suitable ontology editor. Furthermore, the authors hope that by reading this chapter, it will be possible to choose an editing tool for ontology construction according to the project goals. The tools have been described following a list of features. The authors believe that the following features are pertinent: collaborative ontology edition, versioning, graphical tree view, OWL editor and many others (see annex 2).

Keywords: knowledge base; meta model; data sharing; data modeling; knowledge representation; knowledge based system; knowledge sharing; semantic data model; data semantics; ontologies; xml.

Introduction

The World Wide Web is mainly composed of documents written in HTML (Hypertext Markup Language). This language is useful for visual presentation since it is a set of "markup" symbols contained in a Web page intended for display on a Web browser. Humans can read Web pages and understand them, but their inherent meaning is not shown in a way that allows their interpretation by computers. The information on the Web can be defined in a way that can be used by computers not only for display purposes, but also for interoperability and integration between systems and applications (Cardoso, 2005).

"The Semantic Web is not a separate Web but an extension of the current one, in which information is given a well-defined meaning, better enabling computers and

people to work in cooperation" (Berners-Lee et al., 2001). The Semantic Web was made through incremental changes by bringing machine readable descriptions to the data and documents already on the Web. In recent times, instead of a Web site comprising a collection of manually constructed HTML pages, server-side applications and database access techniques are used to dynamically create Web pages directly in response to requests from user's browsers. The technologies available to dynamically create Web pages based on databases information were insufficient for requirements of organizations looking for application integration solutions. Business required their heterogeneous systems and applications to communicate in a transactional manner.

Ontologies can be used to increase communication either between humans and computers. An ontology is a shared conceptualization of the world. Ontologies consist of definitional aspects such as high-level schemas and assertional aspects, entities, attributes, interrelationships between entities, domain vocabulary and factual knowledge, all connected in a Semantic manner (Sheth, 2003). They have generally been associated with logical inferencing and recently have begun to be applied to the Semantic Web. Ontologies provide specific tools to organize and provide a useful description of heterogeneous content. The three major uses of ontologies are:

- To assist in communication between humans;
- To achieve interoperability and communication among software systems;
- To improve the design and the quality of software systems.

The most prominent markup language for publishing and sharing data using ontologies on the internet is the Web Ontology Language (OWL, 2004). There are several ontology development tools for domain modeling, for building knowledge base systems, for ontology visualization, for project management or other modeling tasks. Many of the tools are research prototypes that have been built for a particular project or for an Institute/University. There has been a significant growth in the number of ontology technologies products.

After studying Michael Deny's *Survey on Ontology Tools* and reading the paper *The Hitchhiker's Guide to Ontology Editors* of Loredana Laera and Valentina Tamma we decided to do an update of the tools that are available. Some of the tools described in the Michael Deny's Survey either were no longer available (the project has finished) or have been improved. There are also new tools and new languages since there are new projects that demand so. In composing the list shown on table 1, we have selected the tools that comprise some of the following features: are robust and ready to be used; free and open source; provide support to most of the activities involved in the ontology development process and ontology practice; support Resource Description Framework (RDF), Resource Description Framework Schema (RDFS) and Web Ontology Language (OWL); offer collaborative environment; provide multiple ontology environment; offer server-based environment with support for consistency checking; offer easy-to-use functionality for visual creation and editing; offer a query builder; support a methodology; support editing formal axioms and rules; support the growth of large scale ontologies; support versioning; promote interoperability; has a reasoner; has a graphical view; promotes easy and fast navigation between concepts; has tutorial support; and offers Plug-ins.

We have chosen the following tools: Protégé; OntoEdit; DOE (Differential Ontology Editor); IsaViz; Ontolingua; Altova SemanticWorks 2006; OilEd; WebODE; pOWL and SWOOP.

Protégé is one of the most widely used ontology development tool. It is free and open source. It is an intuitive editor for ontologies and there are plug-ins available to carry

out some of the tasks for building an ontology. OntoEdit is an ontology editor that integrates numerous aspects of ontology engineering. OntoEdit environment supports collaborative development of ontologies. DOE is a simple prototype developed with Java that allows users to build ontologies according to the Bruno Bachimont proposed methodology. IsaViz is a visual environment for browsing and authoring RDF models as graphs. Ontolingua was built to ease the development of ontologies with a form-based Webinterface.

Altova Semantic Works is a commercial visual Semantic Web editor that offers easy-to-use functionality for visual creation and editing. It can be downloaded for 30 days free evaluation period. OilEd is an editor that allows the user to construct and manipulate DAML+OIL (DAML- DARPA Agent Markup Language; OIL-Ontology Inference Layer) and OWL ontologies and which uses a reasoner to classify and check consistency of ontologies. It is provided free of charge. WebODE is the web counterpart for ODE (Ontology Design Environment). It has support for multiple-users. This editor gives support to the methodology Methontology. pOWL is an open source ontology management tool in a collaborative Web enabled environment. SWOOP is a Web-based OWL ontology editor and browser. This editor has default plug-ins for different presentation syntax for rendering ontologies.

The main purpose of this chapter is to give an overview of some of the ontology tools available at the present time. This way, if you are starting out on an ontology project, the initial step is to find a suitable ontology editor.

PROTÉGÉ

Protégé (Noy et al., 2001) is one of the most widely used ontology development tool that was developed at Stanford University. Since Protégé is free and open source, it is supported by a large community of active users. It has been used by experts in domains such as medicine and manufacturing for domain modeling and for building knowledge-base systems. Protégé provides an intuitive editor for ontologies and has extensions for ontology visualization, project management, software engineering and other modeling tasks.

In early versions, Protégé only enabled users to build and populate frame-based ontologies in accordance with the Open Knowledge Base Connectivity protocol (OKBC). In this model, an ontology consisted of a set of classes organized in a subsumption hierarchy, a set of slots associated to classes to describe their properties and relationships, and a set of instances of those classes. Protégé editor included support for classes and class hierarchies with multiple inheritance; templates and slots; predefined and arbitrary facets for slots, which included permitted values, cardinality restrictions, default values, and inverse slots; metaclasses and metaclass hierarchy.

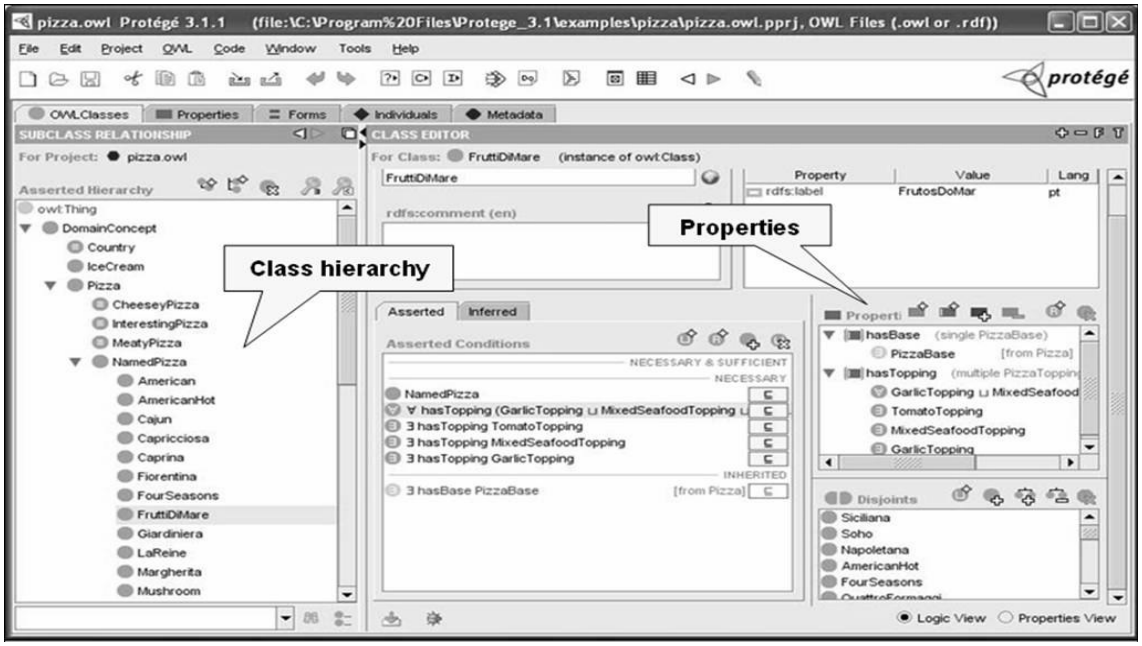

Figure 1: Protégé editor

While the first architecture of Protégé was based on frames, in 2003 it has been extended to support OWL. This extension has attracted many users captivated by the Semantic Web vision. The OWL plug-in extends the Protégé platform into an ontology editor for the OWL enabling users to build ontologies for the Semantic Web. The OWL plug-in allows users to load, save, edit and visualize ontologies in OWL and RDF. It also provides interfaces for Description Logic Reasoners such as Racer.

Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and Extended Mark-up Language (XML) Schema. The current Protégé version can be used to edit classes and their characteristics, to access reasoning engines, to edit and execute queries and rules, to compare ontology versions, to visualize relationships between concepts, and to acquire instances using a configurable graphical user interface. Protégé is a tool installed locally in a computer and does not allow collaborative editing of ontologies by groups of users.

Protégé can be extended by way of a plug-in architecture and a Java-based Application Programming Interface (API) for building knowledge-base tools and applications. Protégé is based on Java and provides an open-source API to develop Semantic Web and knowledge-base stand-alone applications. External Semantic Web applications can use the API to directly access Protégé knowledge bases without running the Protégé application. An OWL API is also available to provide access to OWL ontologies. Its extensible architecture makes it suitable as a base platform for ontology-based research and development projects. Protégé also includes a Programming Development Kit (PDK), an important resource for programmers that describe how to work directly with Protégé APIs and illustrates how to program plug-in extensions for Protégé.

Several plug-ins are available. For example, JSave (http://protege.stanford.edu/plug-ins/jsave/) is an application plug-in to generate Java class definition stubs for Protégé classes and Protégé Web Browser is a Java-based Web application that allows users to share Protégé ontologies over the Internet. The WordNet plug-in (http://protege.stanford.edu/plug-ins/wordnettab/wordnet_tab.html) provides Protégé users an interface to WordNet knowledge base. Users can easily annotate a Protégé

knowledge base using information from WordNet database. The information in WordNet can be searched by name and then be used to annotate ontologies with terms, concept IDs, synonyms, and relations.

The XML Schema (http://faculty.washington.edu/gennari/Protege-plug-ins/XMLBackend/XMLBackend.html) is a backend plug-in that transforms a Protégé knowledge base into XML. The plug-in generates an XML Schema file describing the Protégé knowledge model and an XML file where the classes and instances are stored. The UML plug-in (http://protege.stanford.edu/plug-ins/uml/) is also a backend plug-in which provides an import and export mechanism between the Protégé knowledge model and the object-oriented modeling language UML. To enables the exchange of ontologies and UML class diagrams, the UML plug-in uses the standard format for UML diagram exchange, XMI, which is supported by major CASE tools. The use of the XMI standard enables users to work with Protégé in combination with Software Engineering tools and Integrated Development Environments.

The DataGenie (http://faculty.washington.edu/gennari/Protege-plug-ins/DataGenie/index.html) is an import/export plug-in that allows reading and creating a knowledge model from relational databases using JDBC. Users can select a proper subset of a relational database to be converted into Protégé classes. Typically, during the conversion, tables become classes and attributes becomes slots. The Docgen (http://protege-docgen.sourceforge.net/) is also an import/export plug-in that allow users to create reports describing Protégé knowledge bases or ontologies. Classes, instances and documentation can be exported to various output formats such as HTML, Dynamic Hypertext Markup Language (DHTML), PDF, and XML.

Plug-ins are also available to carry out rule-based programming using the information stored in a Protégé frame-based knowledge base. Two worth mentioning examples are JessTab (http://www.ida.liu.se/~her/JessTab/) and Algernon (http://algernon-j.sourceforge.net/doc/algernon-protege.html). JessTab is a plug-in that provides a Jess console window where it is possible to interact with Jess while running Protégé. This plug-in extends Jess with supplementary features that map Protégé knowledge bases to Jess facts. Users can deploy applications that handle Protégé knowledge bases and react when patterns in the knowledge base are found. Algernon is a system implemented in Java that performs forward and backward inference of frame-based knowledge bases. Compared to Jess, Algernon operates directly on Protégé knowledge bases rather than requiring a mapping operation to and from a separate memory space.

The PROMPT plug-in (Noy and Musen, 2003) allows to manage multiple ontologies within Protégé, mainly compare versions of the same ontology, merge ontologies into one, and extract parts of an ontology.

The OWL-S Editor plug-in (http://owlseditor.semwebcentral.org/) is an easy-to-use editor which allows loading, creating, managing, and visualizing OWL-S services. OWL-S (formerly DAML-S) is emerging as a Web service description language that semantically describes Web Services using OWL ontologies. OWL-S consists of three parts expressed with OWL ontologies: the service profile, the service model, and the service grounding. The profile is used to describe “what a service does”, with advertisement and discovery as its objective. The service model describes “how a service works”, to enable invocation, enactment, composition, monitoring and recovery. Finally, the grounding maps the constructs of the process model onto detailed specifications of message formats and protocols. The OWL-S Editor plug-in provides an excellent overview of the relations between the different OWL-S ontologies which

are shown in an intuitive way in the Graphic User Interface (GUI) and can also be shown as a graph

OntoEdit

OntoEdit (Sure et al., 2002) was developed by the Knowledge Management Group of the AIFB Institute at the University of Karlsruhe. It is an ontology engineering environment which allows creating, browsing, maintaining and managing ontologies. The environment supports the collaborative development of ontologies (Sure et al. 2002). This is archived through its client/server architecture where ontologies are managed in a central server and various clients can access and modify these ontologies. Currently, the successor of OntoEdit is OntoStudio (Figure 2) which is a commercial product based on IBM's development environment Eclipse. It can be downloaded for three months free evaluation period.

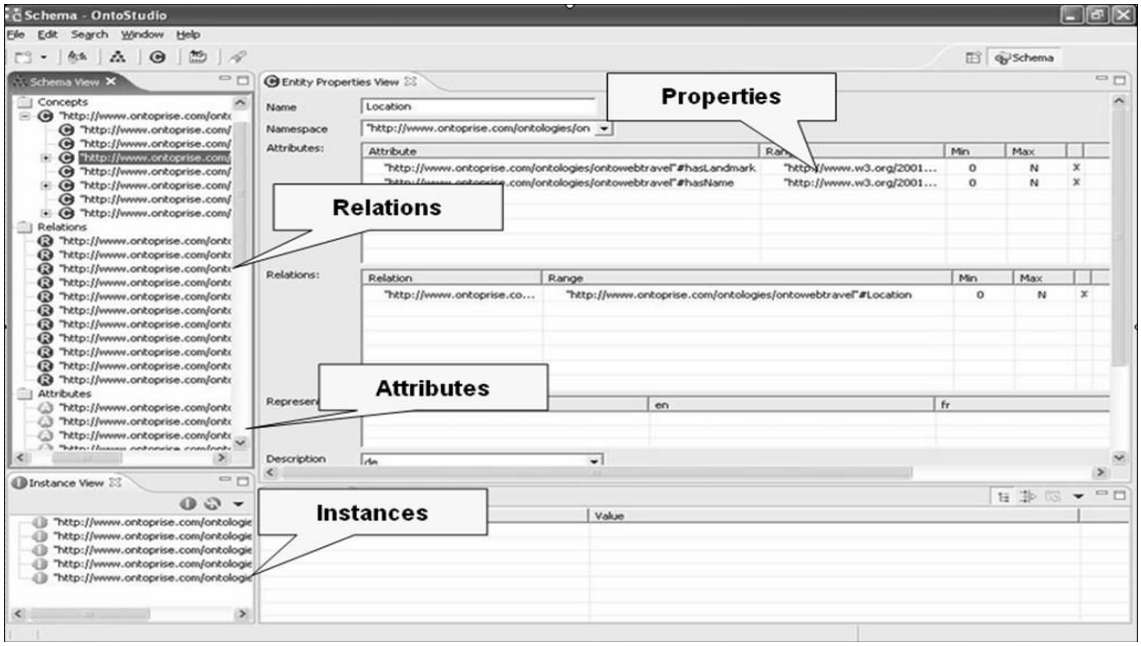

Figure 2: OntoStudio editor

OntoEdit was developed having two major goals in mind. On the one hand, the editor was designed to be as much as possible independent and neutral of a concrete representation language. On the other hand, it was planned to provide a powerful graphical user interface to represent concept hierarchies, relations, domains, ranges, instances and axioms. OntoEdit supports F–Logic (Fuzzy Logic), RDF Schema and OIL. The tool is multilingual. Each concept or relation name can be specified in several languages. This is particularly useful for the collaborative development of ontologies by teams of researchers spread across several countries and speaking different languages. From the technical perspective, this feature is archived by using unique identifiers so that each ontological statement remains clearly defined. The names selected by users serve merely as an external representation.

OntoEdit is built on top of an internal data representation model. The data model of OntoEdit is OXML 2.0 which is frame based. OXML is defined in XML using XML-Schema. Besides concepts, relations and instances, the model can represent predicates, predicate instances, and axioms. Predicates are n-ary associations and are very similar to

predicates defined in first order logic (FOL). Several types of relationships can be established between concepts, such as symmetric, reflexive, transitive,antisymmetric,asymmetric, irreflexive, or intransitive.

The internal representation data model can be exported to DAML+OIL, F-Logic, RDF(S), and OXML. Additionally, ontologies can be exported to relational databases via JDBC. OntoEdit can import external data representation in DAML+OIL, Excel, F-Logic, RDF(S), and OXML. OntoSudio can also import and export OWL files. OntoEdit provides an API for accessing ontologies in an object-oriented fashion. The default API implementation stores ontologies in main-memory, but an additional API exists for persistent storage.

The inference engine that OntoEdit uses is OntoBroker [Decker, 1999]. Using this engine, OntoEdit exploits the strength of F-Logic in that it can represent expressive rules. OntoBroker is the result of several years of research and it is now a commercial product. Like Protégé, OntoEdit is based on a plug-in architecture. The architecture consists of three layers (Figure 3): GUI, OntoEdit core and Parser.

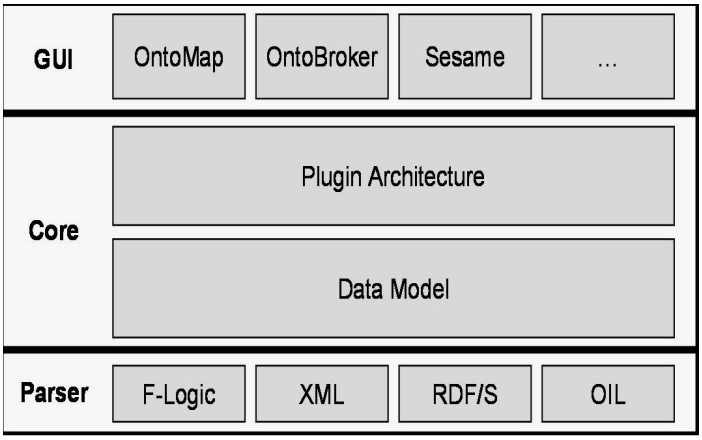

Figure 3: OntoEdit Architecture

Using a plug-in architecture enables users to extend OntoEdit with specific functionalities. Since the plug-in interface is open to third parties, anyone with special requirements can deploy a component to satisfy specific needs. Several plug-ins are available. For example, the Sesame plug-in (Broekstra et al., 2002) is a generic application for storing and querying RDF and RDF Schema. Sesame allows persistent storage of RDF data and schema information. It supplies a useful query engine which supports RQL, an OQL-style query language.

DOE

DOE (Differential Ontology Editor) is a simple ontology editor and was developed by the INA (Institut National de l'Audiovisuel - France). DOE allows users to build ontologies according to the methodology proposed by Bruno Bachimont (Isaac et al., 2002).

DOE has a classical formal specification process. DOE only allows the specification part of the process of structuring ontology. DOE is rather a complement of others editors (DOE, 2006). It is not intended to be a direct competitor with other existing environments (like Protégé, OilEd, OntoEdit or WebODE), instead it was developed to coexist with other editors in a complementary way. This editor offers linguistics-inspired techniques which attach a lexical definition to the concepts and relations used,

and justify their hierarchies from a theoretical, human-understandable point of view (DOE, 2006). Therefore, DOE should be used in combination with another editor. For instance, an editor that has advanced formal specification, e.g. Protégé.

DOE is a simple prototype developed in Java that supports the three steps of the Bruno Bachimont methodology (Isaac et al., 2002). The Bruno Bachimont methodology can be described in the following three steps. In the first step, the user builds taxonomies of concepts and relations. The user has to unambiguously justify the position for each notion in the hierarchy. The user builds a definition, following four principles which come from the *Differential Semantics* theory (Isaac et al, 2002), i.e., 1) Similarity with Parent, 2) Similarity with Siblings, 3) Difference with Sibling and 4) Difference with Parent. For each notion, a meaning and a description has to be given.

Consequently, the user has to explicit state why a notion is similar but more specific than its parent (i.e., Similarity with Parent and Similarity with Siblings), and why this notion is similar but different from its siblings (i.e., Difference with Sibling and Difference with Parent). Hence, every concept is located in a justified and convinced position. It is possible, for the user, to add synonyms and an encyclopedic definition in a few languages for all notions in the Differential Ontology view. The main goal of this step is to reach a semantic agreement about the meaning of the labels used for naming concepts (Isaac et al., 2002).

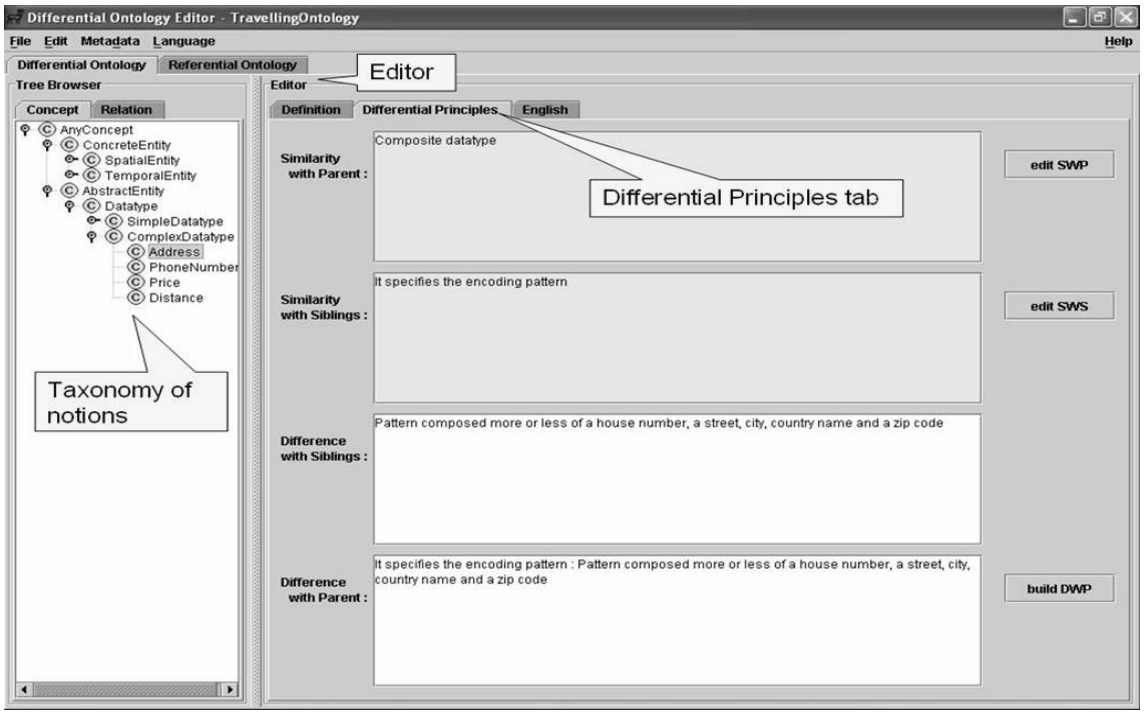

Figure 4: The differential principles bound to the notion addressed in the DOE tool (differential ontology view).

In the second step, the ontological tree obtained in the first step allows to disambiguate the notions. Consequently, the meaning is clarified for a domain-specific application. The notions become concepts behaving as formal primitives. In the referential ontology each concept refers to a set of objects in the domain (its extension) (Isaac et al., 2002). The taxonomies are considered from an extensional semantic point of view. The user can expand them with new entities (defined) or add constraints onto the domains of the relations. In this step, the user ought to do consistency checking in order to look for propagation of the arity all along the hierarchy- if specified - and

inheritance of domains. As we said before, DOE main goal is to guide the user during the first steps of the process of building an ontology. It is not possible to write axioms in DOE since there is not an axiom editor.

Finally, on the third step, the ontology can be translated into a knowledge representation language. The referential concepts are compared with the possible computational operations available in the application Knowledge Based Systems (KBS). As a result, it is possible to use it in an appropriate ontology-based system or to import it into another ontology-building tool to specify it further. An export mechanism was implemented in order to translate the taxonomies into convenient exchange languages (for example, OWL). This mechanism can also be used to complete the third step

DOE allows building a *differential ontology* and a *referential ontology*, corresponding to the first two steps of the methodology proposed by Bruno Bachimont. The Differential Ontology view is divided in two frames. On the left side there is a tree browser and on the right side there is an editor. The tree browser shows hierarchically the concepts and the relations (there is a tab for concepts and a tab for relations too, like in the Referential Ontology view). The editor has three tabs: a definition tab, a different principles tab and an English tab (or other language that the user wishes to translate the ontology to). If the definition tab is selected, it will show the "properties" of the items illustrated in the tree browser. The different principles tab is used to justify the taxonomy of notions that was build. In other words, the sense of a node is specified by the gathering of all similarities and differences attached to the notions found on the way from the root notion (the more generic) to the node that is selected on the tree browser (on the left) (Isaac et al., 2002). In the Referential Ontology view, the tree browser has the same features as in the Differential Ontology view (there are two tabs concept tab and relation tab).

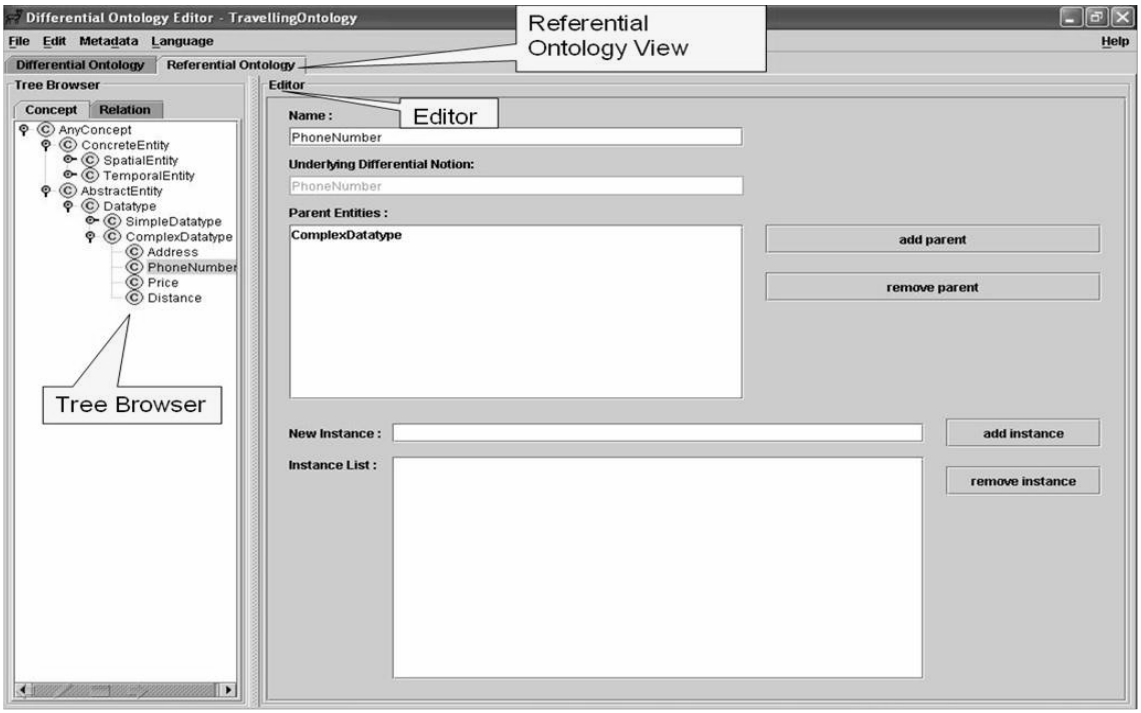

Figure 5: The Referential Ontology view with a concept tree of the Traveling Ontology.

While DOE does not have a graph view of the ontology, it includes a tree view that can show separately the concepts and the relations between the concepts. It is possible to view an ontology in two different ways: a Differential Ontology view and Referential Ontology view.

DOE can import RDFS and Ontology Web Language (OWL) formats. It is possible to export to CGXML, DAML+OIL, OIL (Ontology Inference Language) Plain Text, OIL XML, RDFS, RDF/XML, and OWL. The RDFS import/export feature is very helpful when "exchanging" files between editors, for instance OilEd, Protégé and OntoEdit and vice-versa.

DOE uses XSLT to promote interoperability (Isaac et al., 2003). XSLT is a language for transforming XML documents into other XML documents. All the exchange facilities are done with XSLT transformations. The DOE project does not have a mailing list available to the users. The installation instructions are very simple. There are no plug-ins for this editor and no user manual too.

In DOE Web page (http://opales.ina.fr/public/) there is a self-contained installer of the complete product for Windows. The user has to fill out a form in order to access the download page. The installation is fast. After downloading the product, the user has to run the Setup-DOE-v1.51.exe file and follow the instructions. To run DOE in another platform, it is required to have a Java 2 (TM) Platform, Standard Edition Version 1.3 or later (recommended v1.4.1).

IsaViz

IsaViz is a visual environment for browsing and authoring RDF models as graphs. This tool is offered by W3C Consortium. IsaViz (http://www.w3.org/2001/11/IsaViz/Overview.html) was developed by Emmanuel Pietriga. The first version was developed in collaboration with Xerox Research Centre Europe which also contributed with XVTM, the ancestor of ZVTM (Zoomable Visual Transformation Machine) upon which IsaViz is built. As of October 2004, further developments are handled by INRIA Futurs project In Situ. IsaViz also includes software developed by HP Labs (Jena 2 Semantic Web Toolkit), the Apache Software Foundation (Xerces Java 2), and makes use of the GraphViz library developed by AT&T Research (http://www.w3.org/2001/11/IsaViz/Overview.html).
IsaViz does not follow or include any methodology for building an ontology.

IsaViz imports RDF/XML and N-Triples, and exports RDF/XML, N-Triples, Portable Network Graphics (PNG) and Scalable Vector Graphics (SVG). Therefore, it is possible to import ontologies to other editors, for instance, Protégé or OilEd. The IsaViz environment is composed of four main windows: the IsaViz RDF Editor window, the Graph window, the Definition window and the Attribute window.

- The IsaViz RDF Editor window contains the main menus and a palette of tools.

The Graph window is a ZVTM view (http://www.w3.org/2001/11/IsaViz/Overview.html) in which is displayed the RDF model represented as a 2D graph. ZVTM views display an area of an endless space as seen through a camera. This camera can be moved in the virtual space and its altitude can be changed, resulting in a 2.5D Graphical User Interface with smooth zooming capabilities. IsaViz has a user friendly interface. IsaViz has three different ways of viewing a graph. This can be a distinguishing feature when evaluating this tool with others tools. There are 3 different ways of viewing a graph: Graph View; Radar View

and Property Browser. IsaViz is also recognized by its Zoomable User interface (ZUI). This interface allows rapid navigation of the graphs used to represent the RDF models.

The Definitions window is composed of five tabs:

a) The tab Namespaces is a table containing namespace definitions (with their optional prefix bindings).
b) The tab Property Types is a table containing property type definitions.
c) The tab Property Browser is a text browser in which is exhibit all the properties associated with the currently selected resource.
d) The tab Stylesheets (since version 2.0) is a table containing all Graph Stylesheets (GSS) that should be applied to the model.
e) The tab Quick Input (since version 2.0) is a text area used for pasting statements expressed in RDF/XML, N-Triples or Notation3.

The Attribute window shows the attributes of a selected item of the graph. All the attributes shown in this window can be edited.

IsaViz can render RDF graphs using GSS, a stylesheet language derived from Cascade Style Sheet (CSS) and Scalable Vector Graphics (SVG) for styling RDF models represented as node-link diagrams. Resources and literals are the nodes of the graph (ellipses and rectangles respectively), with properties represented as the edges linking these nodes.

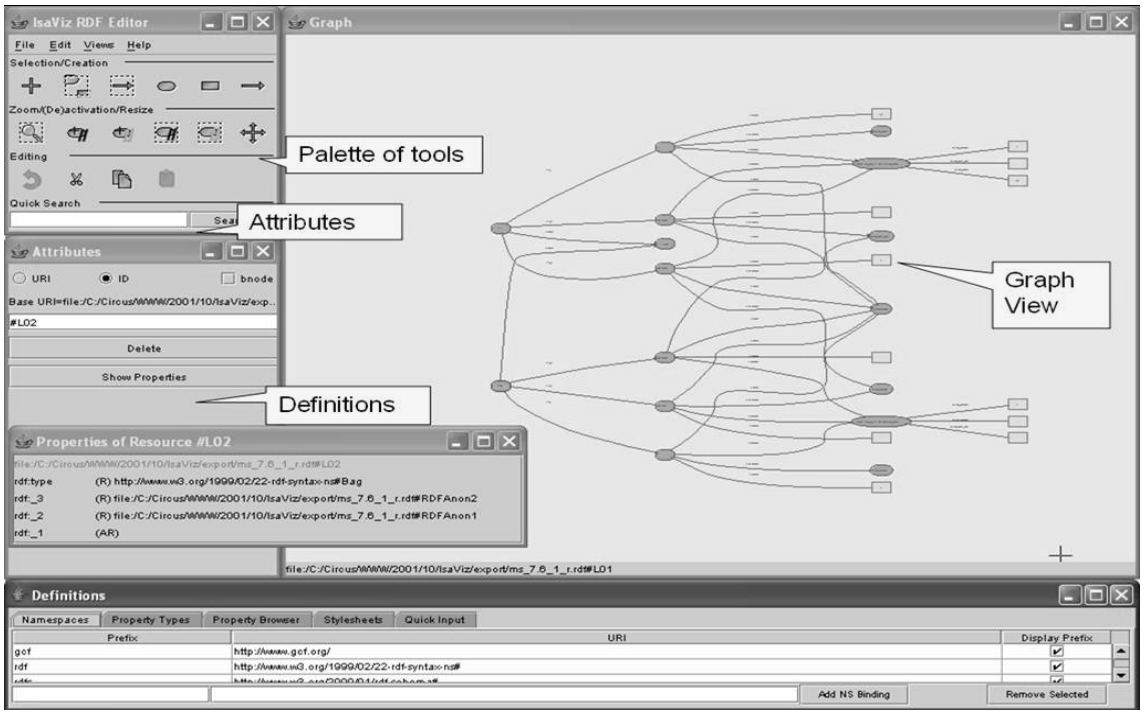

Figure 6: IsaViz window view with a node selected.

This editor has a user manual and a Web page with installation instructions. It also has a mailing list and a list of the most common problems. For that reason, it is really simple to start using this tool. The user can easily solve the problems that can emerge during installation or usage, by using the mailing list of IsaViz.

The current stable version is 2.1 (October 2004) and it is being developed an alpha version: 3.0 (December 2005). IsaViz is implemented in Java and requires a JVM 1.3.x

or later to run (1.4.0 or later is strongly recommended), Jena 1.2 (or later), GraphViz 1.8.x, Xerces 1.4.x, as well as GraphViz (http://www.w3.org/2001/11/IsaViz/Overview.html) for some features. Installation instructions are available at the editor's web page (http://www.w3.org/2001/11/IsaViz/Overview.html).

Ontolingua

The Ontolingua server was the first ontology tool created by the Knowledge Systems Laboratory at Stanford University. Ontolingua was built to ease the development of ontologies with a form-based Web interface. Initially the main application inside the Ontolingua Server was an ontology editor. The Ontology Editor is a tool that supports distributed, collaborative editing, browsing and creation of Ontolingua ontologies. Other systems were included in the environment, such as Webster, Open Knowledge Base Connectivity (OKBC) Server and Ontology merge tool. The Ontolingua Server ontology editor is also known as the Ontolingua Server frame-editor or even as the Ontolingua Server.

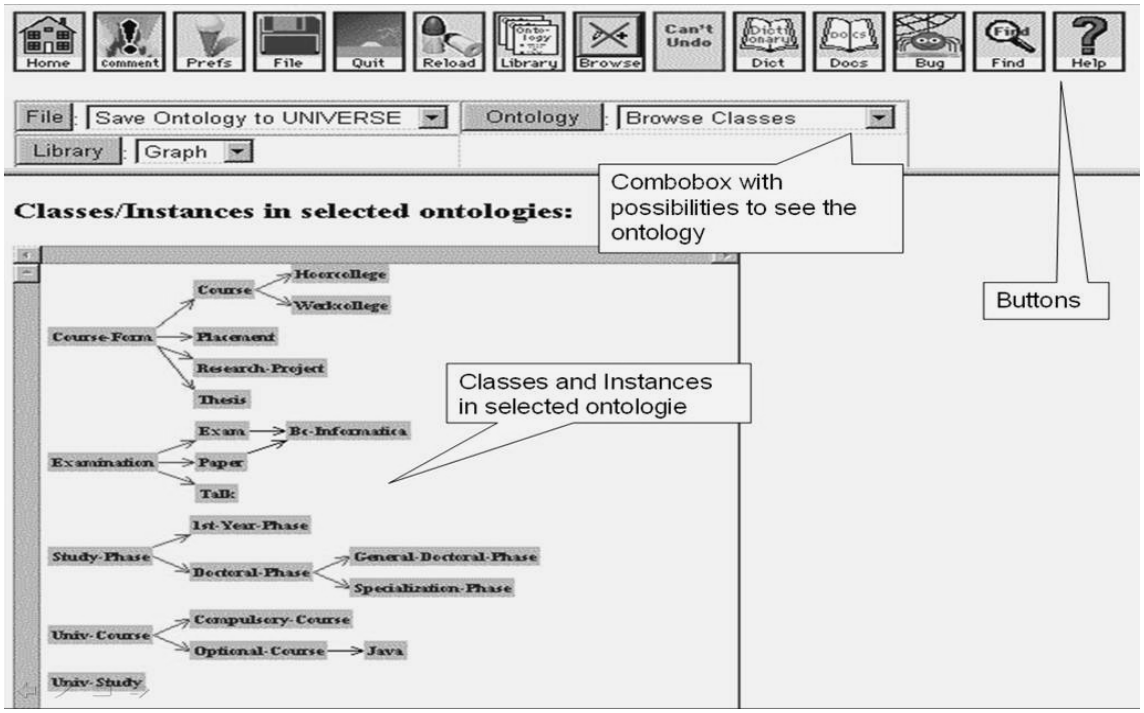

Figure 7 Ontolingua Editor view.

It is possible to export or import to the following formats: DAML+OIL, Knowledge Interchange Format (KIF), Open Knowledge Base Connectivity (OKBC), LOOM, Prolog, Ontolingua, C Language Integrated Production System (CLIPS). It is also possible to import Classic Ocelot and Protégé knowledge bases, but it is not possible to export to these formats.

If an ontology is available in the Ontolingua Server, there are two ways of using it from an external application: connecting to the OKBC Server from a remote client or using the ontology editor to translate the ontology into an implementation language.

The Ontolingua Server is organized as a set of ontology related Web applications, which are built on top of the Ontolingua Knowledge Representation (KR) System.

Ontolingua uses an OKBC model with full KIF axioms. The base language is Ontolingua. This language consists of the combination of frames and first order logic, and allows representing classes. Classes are organized in taxonomies, relations, functions, formal axioms, and instances. The ontology editor is divided in two frames. The frame at the top shows the menu with the editing options available to users. The other frame shows the form to create the new class, where the user must specify the class name, and optionally the natural language documentation and superclasses of the concept. After the form is filled, the user must send its description to the server to store the definition in Ontolingua.

A session can be accessed at http://www.ksl.stanford.edu/software/ontolingua/ by any user in the group that owns the session. The other users connected to the session will be notified of the side-effects that are performed. To establish a new group, the user should use the comment button to email the group name that the user wishes to use along with an initial list of members. The Ontolingua Server ontology editor permits to manage users and user groups who can share their ontologies. Users can work in different sessions and with different groups for each session, and then decide which ontologies can be shared with the members of each group (Farquhar et al., 1995).These multi-user sessions encourage collaborative ontology development. Nevertheless, since no locking mechanisms for ontology terms or version management functions exist in the editor, collaboration may sometimes be somewhat limited. The user can export an ontology (or a set of ontologies) as static HTML files to a local site. It is possible to compare ontologies. The contents of an ontology can be compared with any other loaded ontology. Users often want to split an ontology into a set of more easily understood, more tightly focused modules. The server provides direct support for splitting a large ontology into several smaller ontologies that may include each other. The Ontolingua Server allows assembling Ontolingua Ontologies, so that ontologies can be built in a modular way. Users can reuse existing ontologies from a modular structured library. Ontologies can be reused by inclusion, polymorphic refinement, and restriction. The primary mechanism for supporting ontology reuse is through the library of ontologies. This library acts as a central repository for reusable ontologies (Farquhar et al., 1995).

The main difference from other editors is the fact that users must have some notions of KIF and of the Ontolingua language in order to fill in the forms. Hence, if the user does not have any knowledge of KIF and of the Ontolingua language, the user will not be able to create an ontology since there is no help menu to build the KIF expression of an Ontolingua axiom. The interface is not very user friendly since the buttons are large and primitive and the drawings inside the buttons can sometimes be ambiguous.

There is no need for an installation since this editor has Web access. The user has to be registered in order to use the system. After being registered the following services are available: Chimaera, the Ontology Editor and Webster. Chimaera helps users to reorganize taxonomies and resolve name conflicts in a knowledge-base. It is especially useful when merging KBs, but also useful as an ontology browser and ontological sketchpad. The Ontology Editor allows users to browse, create and edit ontologies using a Web browser. Webster is an online version of the Webster Dictionary, i.e., a searchable hypertext dictionary.

Altova SemanticWorks™ 2006

Altova SemanticWorks 2006 is a commercial visual Semantic Web editor which provides powerful, easy-to-use functionality for a visual creation and editing of RDF,

RDF Schema (RDFS), OWL Lite, OWL DL, and OWL Full documents. This editor has an intuitive visual interface and drag-and-drop functionalities. It allows users to visually design Semantic Web instance documents, vocabularies, and ontologies. There is no evident orientation or methodology associated to this tool.

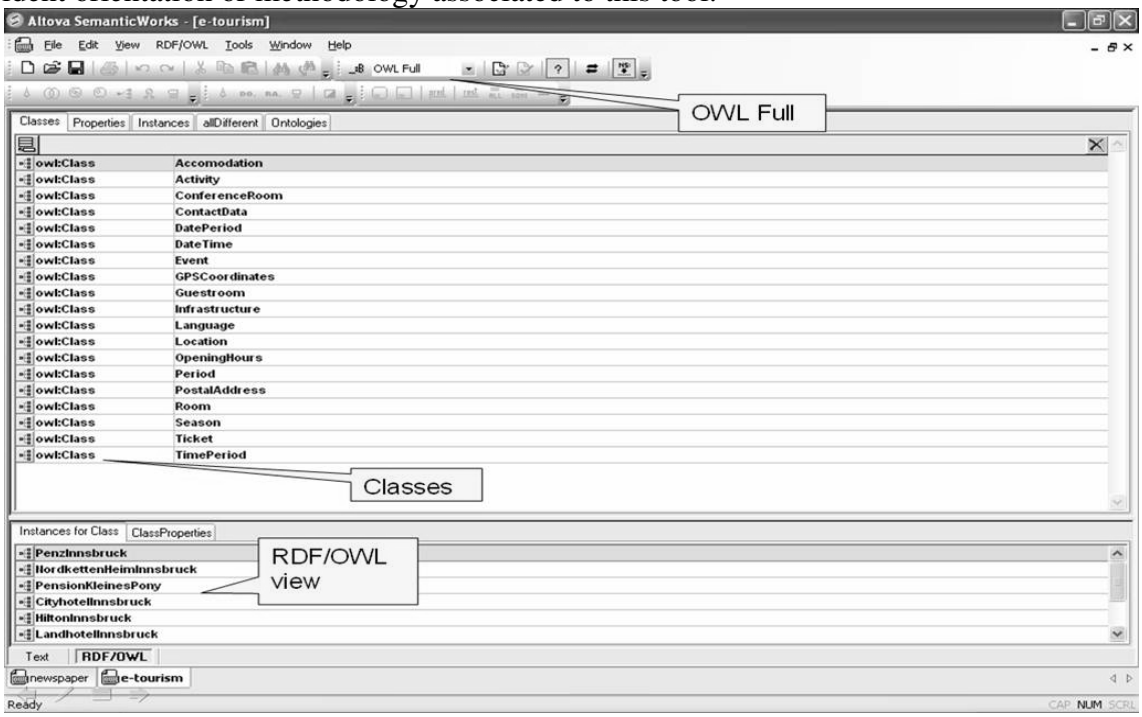

Figure 8 Semantic Works 2006 view bound to the e-tourism ontology.

This editor has the ability to manage the following files: N-triples, XML, OWL, RDF and RDFS. This tool provides powerful features for working with RDF in RDFS vocabularies and OWL ontologies. Users are capable of printing the graphical RDF and OWL representations to create documentation for Semantic Web implementations. The user can switch from the graphical RDF/OWL view to the text view to see how documents are being built in RDF/XML, OWL or N-triples format. The RDF/XML, OWL or N-triples code is automatically generated based on the user's design. Therefore, users can learn and try out with the concepts of the Semantic Web without having to write complicated code.

The graphical display is highly configurable. The user has the possibility to adjust the width of the items in the graph, exhibit them with a vertical or horizontal orientation, adjust the distances between parent and child nodes, and change the font styles and colors used.

An intelligent right menu and context sensitive entry helpers give support to change or add details to the RDF resource according to the user choices. The entry helpers and menus only offer the choices permitted based on the RDF specification, so that users can be certain to create valid documents. Any conflicts are listed in the error window. The errors are written as links so that the user can find and repair them promptly and easily.

A full version of this editor can be installed using a self-contained installer. It is easy and fast to install. The user has to request a free evaluation key, by giving the following information: Name, Company name and e-mail. Users can test this editor for a period of 30 days after receiving the key password by e-mail.

OilEd

OilEd is a graphical ontology editor developed by the University of Manchester for Description Logic Ontologies (Gómes-Pérez et al., 2004). The main purpose of OilEd is to provide a tool for ontologies or schemas editing, as opposed to knowledge acquisition, or the construction of large knowledge base of instances (Laera and Tamma, 2005). OilEd's interface was strongly influenced by Stanford's Protégé toolkit. The initial intention behind OilEd was to provide a simple editor that demonstrated the use of, and stimulated interest in, the Ontology Inference Layer (OIL) language.

The current version of OilEd does not provide a full ontology development environment. OilEd is not capable of supporting the growth of large-scale ontologies, the migration and integration of ontologies, versioning, argumentation and many other activities that are involved in ontology construction. Rather, it is the "Notepad" of the ontology editors, offering enough functionality to allow users to build ontologies and to demonstrate how the Fast Classification of Terminologies (FaCT) reasoner checks ontologies for consistency (Laera and Tamma, 2005). OilEd does not follow any methodology.

OilEd's internal data format is DAML+OIL (DAML- DARPA Agent Markup Language; OIL-Ontology Inference Layer). It is possible to import from DAML+OIL and export ontologies as RDFS, DAML OWL, RDF/XML and others formats.
Figure 7 shows OilEd Editor. The tabs on the main window give access to classes, properties, individuals and axioms. In each tab, a list on the left sides illustrates all the items. On the right panel there is the editor. This editor is used to see or to change the information concerning the entry selected in the item list on the left. The editor was reimplementated reasoner interface so that it was possible to use the DIG (DL Implementation Group) protocol. OilEd includes a DAML+OIL checker. Given the location of an ontology the application checks the syntax of DAML+OIL ontologies and returns a report of the model.

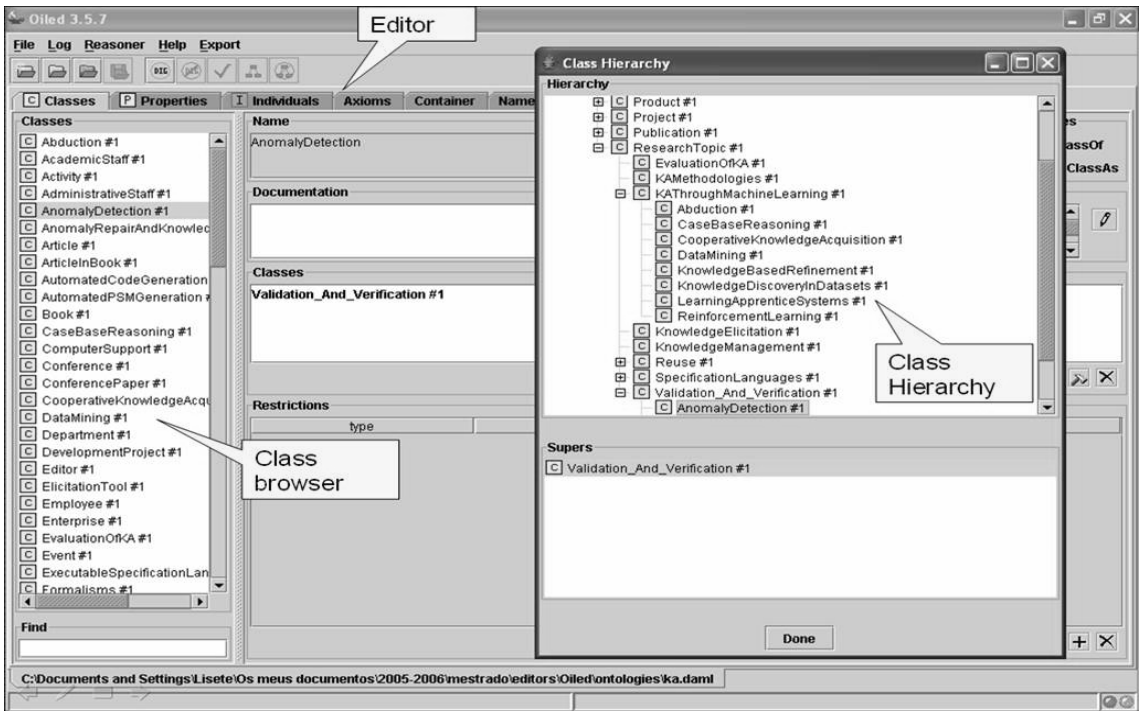

Figure 9 OilEd Ontology editor view bounded with the class AnomalyDetection.

OilEd is provided free of charge, but the user has to provide some information before downloading this editor. Registration is a simple process requiring a valid email address. The user has to be registered, in order to receive a password to revisit the main Web page to download patches and extensions when necessary.

In order to use OWL ontologies, the user must download the sesame-oil.jar library, place it in the /lib directory of the user's installation and remove the existing oil.jar. Sesame is a repository for OWL ontologies. This will add new options to the menus to open and save to Sesame repositories. OilEd expects that these Sesame repositories contain OWL files (rather than, say, DAML+OIL). This only happens since the ontologies are stored in Sesame repositories

The OilEd user manual is included in the distribution which is available as an open-source project under the GPL license. The latest version of OilEd (version 3.5) can be download at http://OilEd.man.ac.uk/. Additionally, Angus Roberts has produced a brief tutorial introduction to OilEd. The following sample ontologies produced using OilEd are available:

- Knowledge Acquisition: The (KA)2 demo ontology from the Ontobroker project.
- Wines: Deborah McGuinness' DAML+OIL wines ontology.
- Diving: An ontology of terms from scuba diving.
- Mad Cows: An ontology demonstrating the use of the reasoner to spot inconsistent concepts.

There are several possible packages for installation. OilEd comes packaged with the FaCT reasoner, although alternative reasoners can be used. OilEd 3.5.7 (Windows)+reasoner. This archive contains a FaCT reasoner for Windows. OilEd 3.5.7 (Linux)+reasoner. This updated version supports export information knowledge base models to OWL-RDF. This archive also contains the FaCT reasoner. OilEd 3.5.7 (no reasoner). This archive does not contain a reasoner. Use this distribution only if you are not interested in using a reasoner or wish to use OilEd with an alternative DIG reasoner.

WebODE

WebODE has been developed by the Ontological Engineering Group (OEG) from the Artificial Intelligence Department of the Computer Science Faculty (FI) from the Technical University of Madrid (UPM). WebODE is an ontological engineering workbench that provides various ontology related services (Laera and Tamma, 2005). It provides support to most of the activities involved in the ontology development process and ontology practice such as ontology edition, navigation, documentation, merge, reasoning, etc.

The WebODE ontology editor is a Web application. It was build on top of the ontology access service (ODE API).

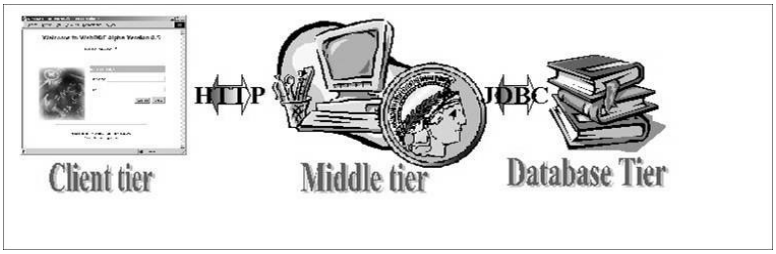

Figure 10 Architecture of the WebODE platform

The WebODE platform has been built using a three-tier model (illustrated in Figure 7) which include:

1) Presentation Tier;
2) Middle Tier;
3) Database Tier.

The first tier (or presentation tier) provides the user interface. This interface is provided by means of a Web browser and uses standard Web technologies. The presentation tier was implemented using HTML (Hyper Text Markup Language), CSS (Cascading Style Sheets) and XML (Extended Mark-up Language), because these technologies permit a simple interoperability between applications. Technologies like JavaScript and Java were used in order to ease the server of the weight of user validations. JavaScript provides easy and quick form validation. Java allows more complex presentation and logic schemas like graphical design or formula validation.

The second tier, the middle tier, provides the business logic. This tier is the result of the combination of two other sub-tiers: presentation sub-tier and logic sub-tier. The presentation sub-tier is in charge of generating the content to be presented in the user's browser. It is also aimed at handling user requests from the client (such as form and query handling) and forwards them to the ODE service as appropriate. Technologies such as servlets or JSPs (Java Server Pages) were used for this purpose. The logic sub-tier provides direct access to ontologies. This direct access is made by means of a well-defined API supplied through an application server (Minerva Application Server). This server provides access to services through RMI (Remote Method Invocation) thus making application development and integration very easy.

The third tier, the database tier, contains the data. Ontologies in WebODE can be stored in any relational database. The database is accessed by means of the JDBC (Java Database Connectivity) standard.

Ontologies in WebODE are conceptualized with a very expressive knowledge model. This knowledge model is based on the reference set of intermediate representations of the Methontology methodology (Laera and Tamma, 2005). The following ontology components are included in the WebODE's knowledge model: concepts and their local attributes (both instance and class attributes), whose type can be any XML Schema type; concept groups, which represent sets of disjoint concepts; concept taxonomies, and disjoint and exhaustive class partitions; ad hoc binary relations between concepts, which may be characterized by relation properties (symmetry, transitiveness, etc); constants; formal axioms, expressed in first order logic; rules; and instances of concepts and relations.

WebODE contains an ontology editor, an ontology-based knowledge management system (ODEKM), an automatic Semantic Web portal generator (ODESeW), a Web resource annotation tool (ODEAnnotate), and a Semantic Web service tool (ODESWS). The ontology editor has three user interfaces: an HTML form-based editor for editing all ontology terms except axioms and rules, a graphical user interface (OntoDesigner)

and a WAB (WebODE Axiom Builder) for editing formal axioms and rules. This editor has the following ontology building services: documentation service, OKBC-based Prolog Inference engine, and ODEClean. WebODE has automatic exportation and importation services from and into XML. WebODE has translation services to other languages and systems or to and from diverse ontology specification languages. Therefore, WebODE presents vast potential capabilities for interoperability.

Ontology export services allow generating WebODE ontologies in XML and in several other ontology languages, such as: RDF(S), OIL, DAML+OIL, OWL, and F-Logic. Ontologies can be transformed and used with the Protégé ontology editor or use the interoperability capabilities provided by this tool. There are several ways of using WebODE ontologies inside ontology-based applications: by its Java API via a local service or application running on the same computer where the ontology server is installed, by generating WebODE ontologies in XML and in several other ontology languages, by transforming ontologies into Protégé-2000 or into Java.

The Java API avoids accessing directly the relational database. WebODE ontologies can be accessed not only from inside the local server but also remotely with RMI and Web services (Gómes-Pérez et al., 2001). It is possible to generate WebODE ontologies into: RDF(S), OIL, DAML+OIL and OWL. The WebODE ontologies can be used inside Protégé-2000 ontology editor.

During this process of transforming ontologies into Java, concepts are transformed into Java beans, attributes into class variables, ad hoc relations into associations between classes. This Java code can then be used to create other Java applications and uploaded in rule systems like Jess.

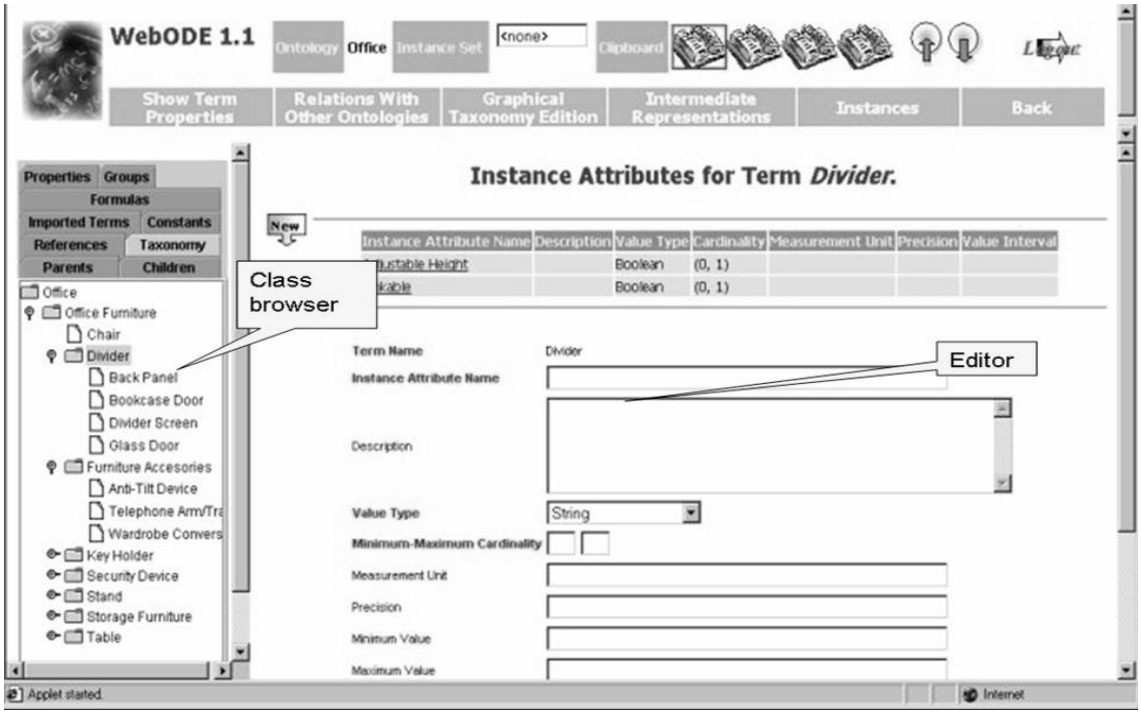

Figure 11 WebODE view bounded to the divider concept.

WebODE has support for multiple-users. User groups can be created to collaborate in the edition of ontologies. Several users can edit the same ontology without errors by means of synchronization mechanisms. One important advantage of using this

application server is that we can decide which users or user groups may access each of the services of the workbench.
This editor provides Web access and therefore, there is no need for any installation activity (http://kw.dia.fi.upm.es/wpbs/#download) .

pOWL

pOWL (Auer, 2005) is a PHP-based open source ontology management tool. pOWL is capable of supporting parsing, storing, querying, manipulation, versioning, serving and serialization of RDFS and OWL knowledge bases in a collaborative Web enabled environment.

pOWL does not follow any specific methodology for developing ontologies. It supports heterogeneous data and its formal description. pOWL tries to follow the W3C Semantic Web Standards. pOWL can import and export model data in different serialization formats such as RDF/XML, and N-Triple.

pOWL is designed to work with ontologies of arbitrary size. This action is limited by the disk space. Therefore, only some parts of the ontology are loaded into main memory. The parts loaded are the ones required to display the information requested by the user on the screen. It offers an RDQL (RDF Data Query language) query builder. pOWL has a tab that correspond to the RDQL query builder. This RDQL is an implementation of an SQL-like query language for RDF. It is possible to query the knowledge base as well as a full-text search for literals and resources. pOWL has an object oriented API. This means that all functionalities are accessed with a clean application programming interface. Models are stored in database tables. It is possible to edit and view models from different points of view.

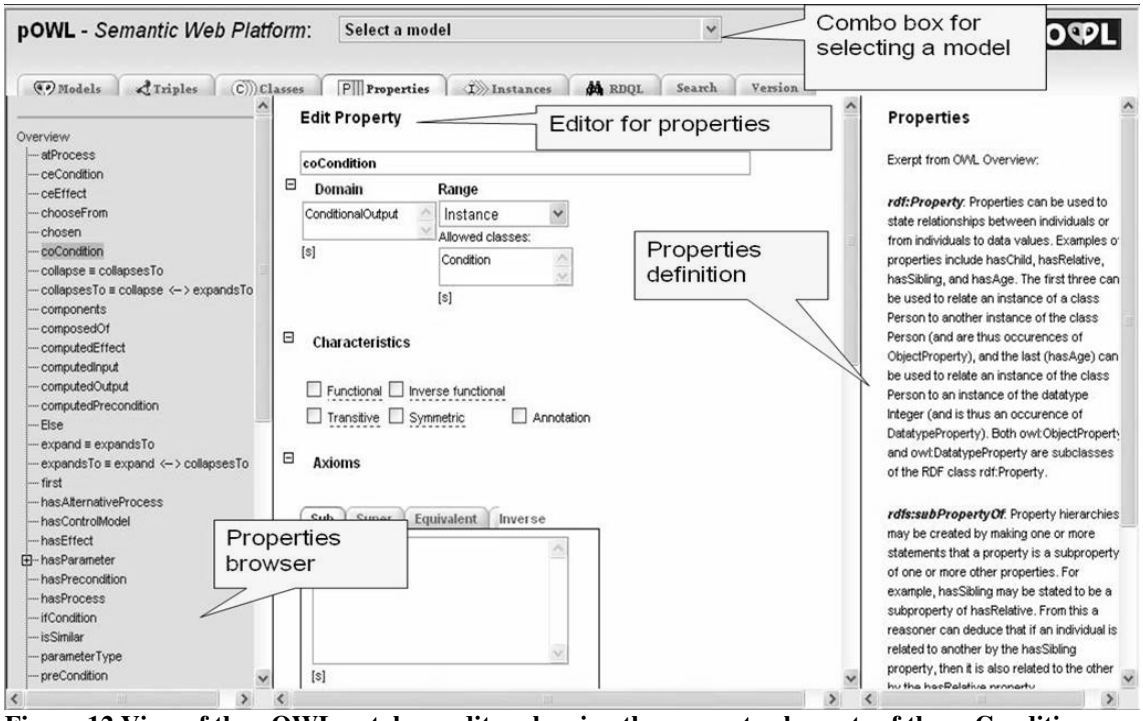

Figure 12 View of the pOWL ontology editor showing the property elements of the coCondition.

pOWL's architecture is formed by four tiers:

a) pOWL store: pOWL store stores its data in a relational database. This database is used to store all the information concerning ontologies and their history.
b) RDFAPI, RDFSAPI, OWLAPI: In order to handle RDF, RDFS and OWL there are three different APIs: RDFAPI, RDFSAPI and OWLAPI. All classes of RDFSAPI are extended with methods for handling OWL predefined properties, Description Logic axioms and restrictions, as well as basic subsumption inference as Lassila and Swick (1999) stated.
c) pOWLAPI: The pOWLAPI includes classes and functions to build Web applications. These web applications are built on top of the APIs already described (Auer, 2005).
d) the user interface: The API that permits accessing, browsing, viewing and editing data models in the pOWL store is the user interface. This interface is based on PHP pages. It is possible to have three different viewpoints of an OWL knowledge base: triple view, database view and a description logic axiom view.

The RDF statements can be viewed on the "triple" view. These statements are written following this principle: subject/predicate/object. The knowledge base can be checked as if it was an object-relational database. OWL classes are represented as tables. The columns of the table represent the properties and the rows represent the instances. On the Description Logic (DL) axiom view, the DL axioms are detected and represented using DL. pOWL does not have a graph view but a tree view is shown on the left side panel (see Figure 9). It has multi language support and has versioning control.

pOWL has the following available documentation: user documentation, installation and administration documentation, developer documentation and application and usage scenarios. pOWL Website has six public areas: an area for bugs, an area for support requests, an area for feature requests, a public forum, a mailing list and a CVS (Concurrent Version System) repository. All edits of a knowledge base may be logged and rolled back. This will only depend on time, user and edit action.

pOWL requires a PHP enabled Web server and a database backend for storing data models. pOWL is available under GNU Public license. The complete source code is also available at http://sourceforge.net/projects/pOWL/.

SWOOP

SWOOP is a Web-based OWL ontology editor and browser. SWOOP contains OWL validation and offers various OWL presentation syntax views. It has reasoning support and provides a multiple ontology environment. Ontologies can be compared, edited and merged. Different ontologies can be compared against their Description Logic-based definitions, associated properties and instances. SWOOP's interface has hyperlinked capabilities so that navigation can be simple and easy. SWOOP does not follow a methodology for ontology construction.

Users can reuse external ontological data (Kalyanpur et al., 2005). This is possible either by purely linking to the external entity, or importing the entire external ontology. It is not possible to do partial imports of OWL. There are several ways to achieve this, such as a brute-force syntactic scheme to copy/paste relevant parts (axioms) of the external ontology, or a more elegant solution that involves partitioning the external ontology while preserving its semantics and then reusing (importing) only the specific partition as desired.

It is possible to search concepts across multiple ontologies. SWOOP makes use of an ontology search algorithm, that combines keywords with DL-based in order to find related concepts. This search is made along all the ontologies stored in the SWOOP knowledge base.

With SWOOP it is possible to have collaborative annotation using the Annotea plug-in. This plug-in presents a useful and efficient Web ontology development. Users may also download annotated changes for a given ontology. The plug-in is used by the users to write and share annotations on any ontological entity. Different SWOOP users can subscribe to the server. Users can maintain different version of the same ontology since mechanisms exist to maintain versioning information using a public server.

SWOOP takes the standard Web browser as the User Interface paradigm. This Web ontology browser has a layout that is well known by most of the users. There is a navigation side bar on the left (Figure 12). The sidebar contains a multiple ontology list and a class/property hierarchy of the ontology. The center pane works like an editor. There is a range of ontology/entity renders for presenting the core content.

This editor provides support for ontology partitioning. OWL ontologies can be automatic portioned into distinct modules each describing a separate domain. There is also support for ontology debugging/repair. SWOOP has the ability to identify the precise axioms that causes errors in an ontology and there are also natural language explanation of the error. An automatic generation of repair plans to resolve all errors are provided. To better understand the class hierarchy, a "CropCircles" visualization format was implemented.

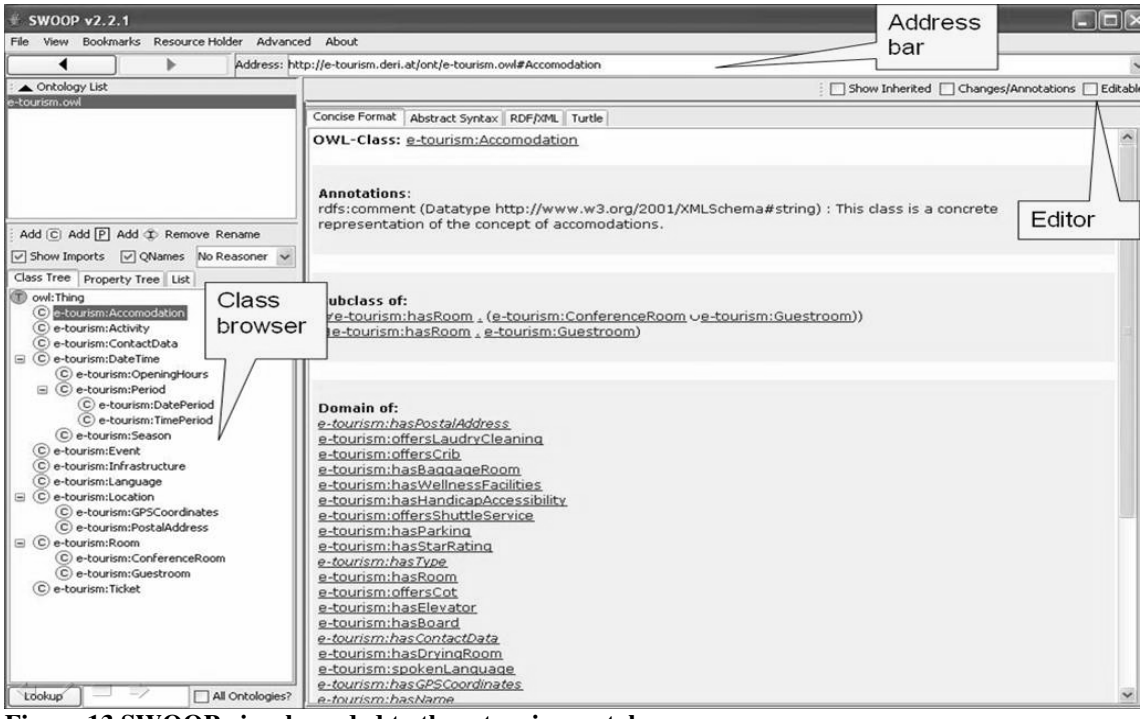

Figure 13 SWOOP view bounded to the e-tourism ontology

SWOOP is based on the conventional Model-View Controller (MVC) paradigm (Figure 11). The SWOOP Model component stores all ontology-centric information pertaining to the SWOOP workspace and defines key parameters used by the SWOOP UI objects. A SWOOP ModelListener is used to reflect changes in the UI based on changes in the SWOOP Model (Kalyanpur et al., 2005). Control is managed through a

plug-in based system. This system loads new Renders and Reasoners dynamically. Therefore, it is possible to guarantee modularity of the code, and encourage external developers to contribute to the SWOOP project easily.

SWOOP uses the Wonder Web OWL API. This API provides programmatic access to data representing OWL ontologies. This API is used as the underlying OWL representation model. It has Ontology Renderers that display statistical information about the ontology, the annotations, the DL expressivity and the OWL species level. There are default plug-ins for different presentation syntax for rendering ontologies. For instance, there are the following presentations syntax RDF/XML, OWL and N3. It is easy to install and has two public mailing lists available: General SWOOP (for users) and Technical SWOOP (for developers). A SWOOP application can be downloaded from http://www.mindswap.org/. After downloading the package, the user has only to run the "runme" batch file.

SWOOP is developed as a separate Java application that attempts to provide the look and feel of a browser-based application. Its architecture was designed to optimize OWL browsing and to be extensible via a plug-in architecture.

Conclusion

Software tools are available to achieve most of the activities of ontology development. Projects often involve solutions using numerous ontologies from external sources. Sometimes there is also the need to use existing and newly developed in-house ontologies. By this reason it is important that the editing tools for ontology construction promote interoperability. As we have stated, Protégé is used for domain modeling and for building knowledge-base systems. Protégé provides an intuitive editor for ontologies and has extensions for ontology visualization, project management, software engineering and other modeling tasks. It also provides interfaces for Description Logic Reasoners such as Racer. Protégé ontologies can be exported into a variety of formats including RDF(S), OWL, and XML Schema. It is a tool installed locally in a computer and does not allow collaborative editing of ontologies by groups of users. There are several plug-ins available for this tool.OntoEdit offers an ontology engineering environment which allows creating, browsing, maintaining and managing ontologies. This editor supports collaborative development of ontologies. The successor of OntoEdit is OntoStudio. OntoEdit supports F–Logic, RDF Schema and OIL. The tool is multilingual. The internal representation data model can be exported to DAML+OIL, F-Logic, RDF(S), and OXML. The inference engine that OntoEdit uses is OntoBroker. OntoEdit is based on a plug-in architecture as Protégé.

DOE allows users to build ontologies according to the methodology proposed by Bruno Bachimont. This editor only permits the specification part of the process of structuring ontology. It should be used in combination with another editor. A *differential ontology* and a *referential ontology* can be built. It is possible to import RDFS and OWL formats and it is possible to export to CGXML, DAML+OIL, OIL Plain Text, OIL XML, RDFS, RDF/XML, and OWL. DOE uses XSLT to promote interoperability.IsaViz is a visual environment for browsing and authoring RDF models as graphs. IsaViz has a user friendly interface and has three different ways of viewing a graph. IsaViz is also recognized by its ZUI. This interface allows rapid navigation of the graphs used to represent the RDF models. IsaViz imports RDF/XML and N-Triples, and exports RDF/XML, N-Triples, PNG and SVG.

Ontolingua was built to ease the development of ontologies with a form-based Web interface. It is a tool that supports distributed, collaborative editing, browsing and creation of Ontolingua ontologies. Ontolingua uses an OKBC model with full KIF axioms. The base language is Ontolingua. It is possible to compare ontologies. Users can reuse existing ontologies from a modular structured library. It is important that users have some notions of KIF and of the Ontolingua language. The interface is not very user friendly. It is possible to export or import to the following formats: DAML+OIL, KIF, OKBC, LOOM, Prolog, Ontolingua, CLIPS.

Altova SemanticWorks is a commercial visual editor that has an intuitive visual interface and drag-and-drop functionalities. Users are capable of printing the graphical RDF and OWL representations to create documentation for Semantic Web implementations. It is possible to switch from the graphical RDF/OWL view to the text view to see how documents are being built in RDF/XML, OWL or N-triples format. The code is automatically generated based on the user's design. This editor has the ability to manage the following files: N-triples, XML, OWL, RDF and RDFS.

OilEd's interface was strongly influenced by Stanford's Protégé toolkit. This editor does not provide a full ontology development environment. However, allows users to build ontologies and to check ontologies for consistency by using FaCT reasoner. OilEd's internal data format is DAML+OIL. It is possible to import from DAML+OIL and export ontologies as RDFS, DAML OWL, RDF/XML and others formats.

WebODE is a Web application. This editor supports ontology edition, navigation, documentation, merge, reasoning and other activities involved in the ontology development process. User groups can be created to collaborate in the edition of ontologies. WebODE has automatic exportation and importation services from and into XML.

pOWL (Auer, 2005) is a PHP-based open source ontology management tool. pOWL is capable of supporting parsing, storing, querying, manipulation, versioning, serving and serialization of RDFS and OWL knowledge bases in a collaborative Web enabled environment. This editor was designed to work with ontologies of arbitrary size. pOWL offers an RDQL query builder. pOWL has versioning control. pOWL supports heterogeneous data and its formal description.

SWOOP is a Web-based OWL ontology editor and browser. SWOOP contains OWL validation and offers various OWL presentation syntax views. It has reasoning support and provides a multiple ontology environment. Ontologies can be compared, edited and merged. It is possible to have collaborative annotation with the Annotea plug-in. Users can reuse external ontological data.

To sum up, there are commercial ontology tools (Semantic Works and OntoStudio), there are ontology tools that demand learning/knowing a specific language (Ontolingua and OilEd) and there are ontology tools that are more graphic (IsaViz and Semantic Works). Other tools are Web-based application (WebODE, pOWL and SWOOP) or follow a methodology (DOE and WebODE).

There are several available ontology tools that can help to build an ontology. Some tools only support common edition and browsing functionalities. Other tools provide ontology documentation, ontology import/export for different formats, graphical view of ontologies, ontology libraries and attached inference engines.

In this chapter, we compared a selection of some of the available ontologies tool and environments that can be used for ontology construction, either from scratch or by reusing other existing ontologies. As we have shown, there are many ontology development tools for ontology construction

References

Auer, S. (2005), *pOWL – A Web Based Platform for Collaborative Semantic Web Development*. In Proceedings of the Workshop on Scripting for the Semantic Web, Heraklion, Greece, May 30, 2005.

Bachimont, B., Isaac, A., & Troncy, R. (2002), *Semantic Commitment for Designing Ontologies: A Proposal*. In Asuncion Gomez-Pérez and V. Richard Benjamins, editors, 13th International Conference on Knowledge Engineering and Knowledge Management, EKAW'2002, volume LNAI 2473, pages 114-121, Sigüenza, Spain, October, 1-4 2002. Springer Verlag.

Bechhofer, S., Horrocks, I., Goble, C., & Stevens, R. (2001), *OilEd: a reason-able ontology editor for the Semantic Web*, Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence.

Berners-Lee, T.J. Hendler, & Lassila, O. (2001), *The Semantic Web.* Scientific American. May 2001.

Broekstra, J., Kampman, A., & Van Harmelen, F. (2002), *Sesame: A generic architecture for storing and querying RDF and RDFSchema*. Horrocks, I., Hendler, J., eds. In Horrocks, I., Hendler, J., eds.: Proc. of the 1st Int. Semantic Web Conf. (ISWC 2002), Sardinia, Italy, LNCS, Springer, Vol. 2342, pp. 54–68.

Cardoso, J. & Amit P. Sheth (2006), *The Semantic Web and its Applications*, in "Semantic Web Services, Processes and Applications". 2006, Springer, ISBN: 0-38730239-5.

Decker, S., Erdmann, M., Fensel, D., & Studer, R. (1999), *Ontobroker: Ontology-based access to distributed and semi-structured information*. In: R. Meersman et al. (eds.), Database Semantics: Semantic Issues in Multimedia Systems, Proceedings TC2/WG 2.6 8th Working Conference on Database Semantics (DS-8), Rotorua, New Zealand, Kluwer Academic Publishers, Boston, 1999. pp. 351–369.

DOE (2006), The Differential Ontology Editor, http://opales.ina.fr/public/index.html

Farquhar, A., Fickas, F., & Rice, J. (1995), *The Ontolingua Server: a tool for collaborative ontology construction* , Proceedings of the 10yh Banff Knowledge Acquisiton for Knowledge based system workshop (KAW 95).

Gómez-Pérez, A. , Fernández-López, M., & Corcho, O. (2004), Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web, pp. 293-362, 2004.

Gómez-Pérez, A., Fernández-López, M., Corcho, O., & Aspiréz, J. (2001), *WebODE: a sacalable ontological engineering workbench*, First International Conference on Knowledge Capture (K-CAP 2001) Canada.

Kalyanpur, A., Parsia, B., & Hendler, J. (2005), *A tool for working with web ontologies*, in In Proceedings of the International Journal on Semantic Web and Information Systems, Vol. 1, No. 1, Jan - March, 2005. URL http://www.mindswap.org/papers/Swoop-Journal.pdf.

Laera, L., & Tamma, V. (2005), T*he Hitchhhiker´s Guide to Ontology Editors*. This paper is based on and updates the extensive evaluation of ontology editors produced by the Ontoweb project in the Deliverable 1.3, "A survey on ontology tools" and can be found at http://ontoweb.aifb.uni-karlsruhe.de/About/Deliverables/D13_v1-0.zip

Lassila, O., & Swick, R.R. (1999), *Resource description framework (RDF) Model and syntax specification*. Recommendation, W3C, February 1999. http://www.w3.org/TR/1999/RECrdf-syntax-19990222

Noy, N.F., Sintek, M., Decker, S., Crubezy, M., Fergerson, R.W., & Musen, M.A. (2001), *Creating Semantic Web Contents with Protege-2000*. IEEE Intelligent Systems, 16, 60--71.

Noy, N.F., & Musen, M.A. (2003), *The PROMPT Suite: Interactive Tools For Ontology Merging And Mapping*, International Journal of Human-Computer Studies, 2003.

Oldakowski, R., & Bizer, C. (2004), *RAP: RDF API for PHP*, First International Workshop on Interpreted Languages, 2004.

Sheth, A. (2003), *Semantic Metadata for Enterprise Information Integration*. DM Review. July 2003.

Sure, Y., Angele, J., & Staab, S. (2002), *OntoEdit: Guiding Ontology Development by Methodology and Inferencing*, In Proceedings of the International Conference on Ontologies, Databases and Applications of Semantics ODBASE 2002, October 28 - November 1, 2002, University of California, Irvine, USA, volume 2519 of LNCS, pp. 1205-1222. 2002.

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer R., & Wenke, D. ,(2002), *OntoEdit: Collaborative Ontology Development for the Semantic* , International Semantic Web Conference, June 9-12 2002, Sardinia, Italy, LNCS, Springer, Vol. 2342, pp. 221–235. 2002.

Troncy, R. & Isaac, A. (2002), *Semantic Commitment for Designing Ontologies: A Tool* Proposal. Poster Session at: 1st International Conference on the Semantic Web, ISWC'2002, Sardinia, Italia, June, 9-12 2002.

Annex 1

Tool	Product or Project Web Page
Apollo	http://apollo.open.ac.uk/index.html
CoGITaNT	http://cogitant.sourceforge.net/
DAMLImp (API)	http://codip.grci.com/Tools/Components.html
Differential Ontology Editor (DOE)	http://opales.ina.fr/public/
Disciple Learning Agent Shell	http://lalab.gmu.edu/
DUET	http://codip.grci.com/Tools/Tools.html
Enterprise Semantic Platform (ESP) including Knowledge Toolkit	http://www.semagix.com/
GALEN Case Environment (GCE)	http://www.kermanog.com/
ICOM	http://www.cs.man.ac.uk/~franconi/icom/
Integrated Ontology Development Environment	http://www.ontologyworks.com/
IsaViz	http://www.w3.org/2001/11/IsaViz/
JOE	http://www.cse.sc.edu/research/cit/demos/java/joe/
KAON (including OIModeller)	http://kaon.semanticweb.org/
KBE -- Knowledge Base Editor (for Zeus AgentBuilding Toolkit)	http://www.isis.vanderbilt.edu/Projects/micants/Tech/Demos/KBE/
LegendBurster Ontology Editor	http://www.georeferenceonline.com/
LinKFactory Workbench	http://www.landc.be/
Medius Visual Ontology Modeler	http://www.sandsoft.com/products.html
NeoClassic	http://www-out.bell-labs.com/project/classic/
OilEd	http://oiled.man.ac.uk/
Onto-Builder	http://ontology.univ-savoie.fr/
OntoEdit	http://www.ontoprise.de/com/ontoedit.htm
Ontolingua with Chimaera	http://www.ksl.stanford.edu/software/ontolingua/
Ontopia Knowledge Suite	http://www.ontopia.net/solutions/products.html
Ontosaurus	http://www.isi.edu/isd/ontosaurus.html
OntoTerm	http://www.ontoterm.com/
OpenCyc Knowledge Server	http://www.opencyc.org/
OpenKnoMe	http://www.topthing.com/
PC Pack 4	http://www.epistemics.co.uk/
Protégé-2000	http://protege.stanford.edu/index.html
RDFAuthor	http://rdfweb.org/people/damian/RDFAuthor/
RDFedt	http://www.jan-winkler.de/dev/e_rdfe.htm
SemTalk	http://www.semtalk.com/
Specware	http://www.specware.org/
Taxonomy Builder	http://www.semansys.com/about_composer.html
TOPKAT	http://www.aiai.ed.ac.uk/~jkk/topkat.html
WebKB	http://meganesia.int.gu.edu.au/~phmartin/WebKB/doc/generalDoc.html
WebODE	http://delicias.dia.fi.upm.es/webODE/
WebOnto	http://kmi.open.ac.uk/projects/webonto/

Table 1: List of some of the representative ontology tools

Annex 2

The following table (Table 2) lists the features that we considered more important when deciding which ontology tool to use: versioning; collaborative ontology edition; graphical class/properties taxonomy; graphical tree view; support the growth of large scale ontologies; querying; friendly user interface; consistency check and OWL editor. The feature versioning keeps track of the version evolution that the ontology suffers. The collaborative ontology edition is a very useful feature since it allows users to edit ontologies in a collaborative manner. The graphical class/properties taxonomy provides an interesting view of the classes and of the properties of the ontology. This feature permits viewing classes and properties with graphically. The graphical tree view shows a generic graph view of the ontology. It is possible to have an overview of the structure of the ontology. If the ontology editor supports the growth of large scale ontologies, then the user can be sure that it is possible to build an ontology by using only one editor. The feature concerning querying allows querying the knowledge base. It is important that the user interface is friendly and similar to others interface. As a result, the user does not need to spend too much time getting to know the tool. Consistency checking is important in order to look for propagation of the arity all along the hierarchy and inheritance of domains.

Features	**DOE**	**IsaViz**	**Ontolingua**	**Semantic Works 2006**	**OilED**	**SWOOP**	**pOWL**	**WebODE**
Versioning	×	×	×	×	×	✓	×	×
Collaborative ontology edition	×	×	✓	×	×	✓	✓	✓
Graphical class/properties taxonomy	✓	×	✓	✓	✓	✓	✓	✓
Graphical tree view	×	✓	✓	✓	✓	×	×	✓
Support growth of large scale ontologies	×	✓	✓	✓	×	✓	✓	✓
Querying	×	×	×	×	×	×	✓	×
User Interface (multi-language, intuitive)	×	✓	×	✓	✓	✓	✓	✓
Consistency Check	×	×	×	× (Syntax)	✓	×		✓
OWL Editor	✓	×	✓	✓	✓	✓	✓	✓

Table 2: Most Representative Features