WebODE: a Scalable Workbench for Ontological Engineering

Julio C. Arpírez, Oscar Corcho, Mariano Fernández-López, Asunción Gómez-Pérez
Facultad de Informática. Universidad Politécnica de Madrid
Campus de Montegancedo, s/n. 28660 Boadilla del Monte. Madrid. Spain
+34 91 336 7439
jarpirez@delicias.dia.fi.upm.es; {ocorcho, mfernandez, asun}@fi.upm.es

ABSTRACT
This paper presents WebODE as a workbench for ontological engineering that not only allows the collaborative edition of ontologies at the knowledge level, but also provides a scalable architecture for the development of other ontology development tools and ontology-based applications. First, we will describe the knowledge model of WebODE, which has been mainly extracted and improved from the reference model of METHONTOLOGY's intermediate representations. Later, we will present its architecture, together with the main functionalities of the WebODE ontology editor, such as its import/export service, translation services, ontology browser, inference engine and axiom generator, and some services that have been integrated in the workbench: WebPicker, OntoMerge and the OntoCatalogue.

Keywords
WebODE, ontology engineering workbench, ontology building, translation, integration and merge.

1. INTRODUCTION

In the last years, several tools for building ontologies have been developed: Ontolingua [11], OntoSaurus [24], WebOnto [8], Protégé2000 [25], OilEd [20], OntoEdit [21], etc. A study comparing some of them can be found in [10]. Additional ontology tools and services have been built for other purposes: ontology merging (Chimaera [18], Ontomorph [4], PROMPT [14]), ontology access (OKBC [5]), etc. Finally, many applications have been built upon ontologies: Ontobroker [12], PlanetOnto [9], $(KA)^2$ [1], MKBEEM [19], etc. All these tools and applications have contributed to a high development of the ontology community, and have laid the foundations of an emergent research and technological area: the Semantic Web [2].

However, current ontological technology suffers from the following problems, which must be solved prior to its transfer to the enterprise world:

- There is no correspondence between existing methodologies and environments for building ontologies, except ODE and METHONTOLOGY [13].
- Existing environments just give support for designing and implementing ontologies, but they do not support all the activities of the ontology life cycle.
- There are a lot of isolated ontology development tools that cannot interoperate easily, because they are based on different technologies, on different knowledge models for representing ontologies, etc.

Consequently, there is a need for a common workbench to ensure a wide acceptance and use of ontological technology. We foresee three main areas in this workbench, as shown in figure 1:

- Ontology development and management, which comprises technology that gives support to ontology development activities: knowledge acquisition, edition, browsing, integration, merging, reengineering, evaluation, implementation, etc.; ontology management activities: configuration management, ontology evolution, ontology libraries, etc.; and ontology support activities: scheduling, documentation, etc.
- Ontology middleware services, which include different kinds of services that will allow the easy use and integration of ontological technology into existing and future information systems, such as services for accessing ontologies, integration with databases, ontology upgrading, query services, etc.
- Ontology-based applications development suites, which will allow the rapid development and integration of ontology-based applications. They will be the last step towards a real integration of ontologies into enterprise information systems.

In this paper, we will present WebODE as an scalable ontological engineering workbench that gives support to activities from the first two areas of the workbench previously identified. WebODE's ontology editor allows the collaborative edition of ontologies at the knowledge level, supporting the conceptualization phase of METHONTOLOGY and most of the activities of the ontology's life cycle (reengineering, conceptualization,

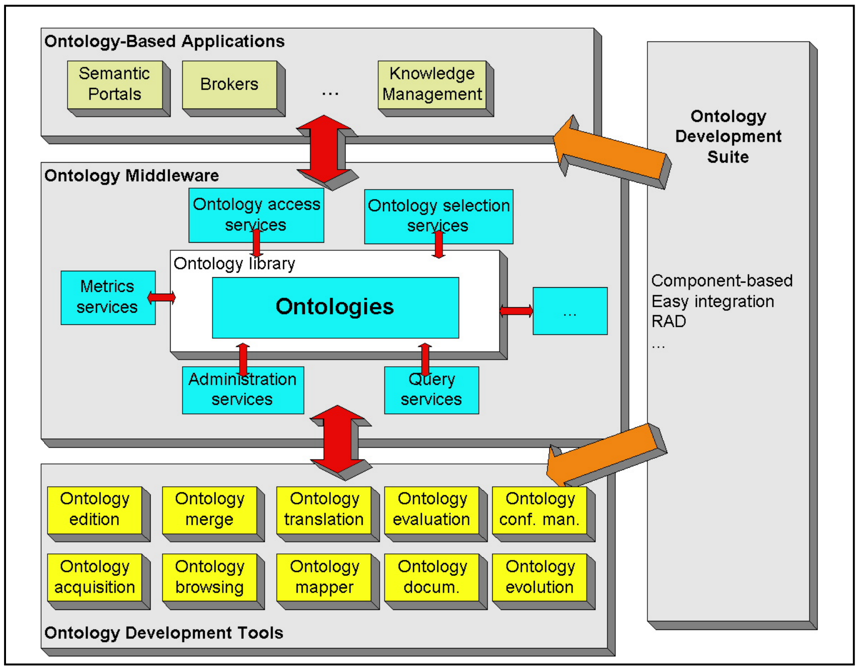


Figure 1. An ontological engineering workbench.

implementation, etc). Besides, WebODE provides high extensibility in an application server basis, allowing the creation of middleware services that will allow the use of ontologies from applications.

This paper is organized as follows: section *WebODE in a nutshell* gives a general overview of the main features of this ontological engineering workbench. Section *WebODE's knowledge model* presents the knowledge model used for representing ontologies in the WebODE workbench. Section *WebODE architecture* describes its main services and the WebODE ontology editor, as an applications that uses most of the services. Section *Related Work* gives a short overview of existing ontology editing applications. Finally, the *Conclusions* section summarizes the main conclusions of this work, projects in which WebODE has already been used, ontologies developed using the WebODE ontology editor and further work.

2. WEBODE IN A NUTSHELL

WebODE is not an isolated tool for the development of ontologies, but an advanced ontological engineering workbench that provides varied ontology related services and covers and gives support to most of the activities involved in the ontology development process.

WebODE workbench is built on an application server basis, which provides high extensibility and usability by allowing the addition of new services and the use of existing services. Examples of these services are *WebPicker*, *OntoMerge* and *OntoCatalogue*.

Ontologies in WebODE are stored in a relational database. Moreover, WebODE provides a well-defined service-oriented API for ontology access that makes it easy the integration with other systems.

Ontologies built with WebODE can be easily integrated with other systems by using its automatic exportation and importation services from and into XML, and its translation services into and from varied ontology specification languages (currently, RDF(S) [23], OIL [16], DAML+OIL [7], X-CARIN [19] and FLogic [17]).

WebODE's ontology editor allows the collaborative edition of ontologies at the knowledge level. Its knowledge model, which is described in depth in the next section, is mainly based on the set of intermediate representations of METHONTOLOGY and provides additional features.

Ontology edition is aided both by form based and graphical user interfaces, a user-defined-views manager, a consistency checker, an inference engine, an axiom builder and the documentation service.

Two interesting and novel features of WebODE with respect to other ontology engineering tools are: instance sets, which allow to instantiate the same conceptual model for different scenarios, and conceptual views from the same conceptual model, which allow creating and storing different parts of the ontology, highlighting and/or customizing the visualization of the ontology for each user.

The graphical user interface allows browsing all the relationships defined on the ontology as well as graphical-pruning these views with respect to selected types of relationships. Mathematical properties such as reflexive, symmetric, etc. and other user-defined properties can be also attached to the "ad hoc" relationships.

The collaborative edition of ontologies is ensured by a mechanism that allows users to establish the type of access of the ontologies developed, through the notion of groups of users. Synchronization mechanisms also exist that allow several users to edit the same ontology without errors.

Constraint checking capabilities are also provided for type constraints, numerical values constraints, cardinality constraints and taxonomic consistency verification [15] (i.e., common instances of disjoint classes, loops, etc.)

Finally, WebODE's inference service has been developed in Ciao Prolog. Although WebODE is not OKBC compliant yet, all the OKBC primitives have been defined in prolog for their use in its inference engine.

3. WEBODE'S KNOWLEDGE MODEL

WebODE's knowledge model is extracted from the set of intermediate representations of METHONTOLOGY. It allows the representation of concepts and their attributes (both class and instance attributes), taxonomies of concepts, disjoint and exhaustive class partitions, ad-hoc binary relations between concepts, properties of relations, constants, axioms and instances. It also allows the inclusion of bibliographic references for any of them and the importation of terms from other ontologies.

Additionally, WebODE improves the reusability of ontologies defining sets of instances, which allow the instantiation of the same conceptual model for different scenarios it may be used for.

In the following subsections we will describe each one of the components of the *WebODE's* knowledge model:

3.1. Concepts

In short, a concept (also known as a class) can be anything about which something is said, and, therefore, can also be the description of a task, function, action, strategy, reasoning process, etc.

Concepts are identified by their **name**, although they can also have **synonyms** and **abbreviations** attached to them. A natural language (NL) **description** can be also included.

The same applies to **references** and **formulae**, which will be described later in this section. Any component in WebODE may have any amount of references and reasoning formulae attached to it.

Class attributes are attributes whose value must be the same for all instances of the concept. They are not components themselves in WebODE's knowledge model, as they are always attached to a concept (and to its subclasses, because of the inheritance mechanism).

The information stored for a class attribute is the following: its **name** (which must be different from the rest of attribute names of the same concept); the **name of the concept** it belongs to (attributes are local to concepts, that is, two different concepts can have attributes with the same name); its **value type**, also called range, which can be a basic data type (String, Integer, Cardinal, Float, Boolean, Date, Numeric Range, Enumerated, URL) or an instance of a concept (in this case, the name of the concept must be specified), and, finally, its **minimum and maximum cardinality**, which constrains the number of values that the class attribute may have.

Optional information for class attributes consists of its **NL description**, the **measurement unit** and its **precision** (the last two ones just in case of numeric attributes).

Finally, the **value(s)** of the class attribute can be specified once it has been defined completely. These values will be attached to the class attribute where they have been defined.

Instance attributes are attributes whose value may be different for each instance of the concept. They have the same properties than class attributes and two additional properties, **minimum value** and **maximum value**, which are used in attributes with numeric value types. Values inserted for instance attributes are interpreted as default values for them.

3.2. Groups

Groups, also called partitions, are used to create disjoint and exhaustive class partitions. They are sets of disjoint concepts that have a **name**, the **set of concepts** they group together and, optionally, a **NL description**. A concept can belong to several groups.

3.3. Built-in Relations

This subsection deals with predefined relations in the WebODE's knowledge model, related to the representation of taxonomies of concepts and mereology relationships between concepts. They are divided into three groups:

Taxonomical relations between concepts. Two predefined relations are included: *subclass-of* and *not-subclass-of*. Single and multiple inheritance are allowed.

Taxonomical relations between groups and concepts. A group is a set of disjoint concepts. There are two predefined relations available, whose semantics is also explained:

- *Disjoint-subclass-partition*. A disjoint subclass partition Y of class X defines the set Y of disjoint classes as subclasses of class X. This classification is not necessarily complete: there may be instances of X that are not included in any subclass of the partition.
- *Exhaustive-subclass-partition*. An exhaustive subclass partition Y of class X defines the set Y of disjoint subclasses as subclasses of the class X, where X can be defined as union of all the classes of the partition

Mereological relations between concepts. Two relations are included: *transitive-part-of* and *intransitive-part-of*.

3.4. Ad-hoc relations

WebODE allows just binary ad-hoc relations to be created between concepts. The creation of relations of higher arity must be made by reification (creating a concept for the relation itself and *n* binary relations between the concepts that appear in the relation and the concept that is used for representing the relation).

Ad-hoc relations are characterized by their **name**, the name of the **origin** (source) and **destination** (target) concepts, and its **cardinality**, which establishes the number of facts (instances of the relation) that can hold between the origin and the destination term. Their cardinality can be restricted to 1 (only one fact) or N (any number of facts).

Additionally, there is some optional information that can be provided for an ad-hoc relation, such as its **NL description** and its **properties** (they are used to describe algebraic properties of the relation).

References and formulae can be also attached to the ad-hoc relations, as happened with the concepts.

3.5. Constants

Constants are components that have always the same value. They are included in the knowledge model of WebODE to ease the maintenance of ontologies. They are available for their use in any expression in the ontology.

The information needed for a constant is: **name**, **value type** (the same as shown for attributes of concepts, except for instances of concepts), **value** and **measurement unit**. Its **NL description** can be optionally provided.

3.6. Formulae

There are three types of formulae that can be created in WebODE: axioms, rules and procedures. All of them are represented by their **name**, an optional **NL description** and a **formal expression** in first order logic, using a syntax provided by WebODE.

Axioms model sentences, using first order logic, that are always true. They may be included in the ontologies for several purposes, such as constraining its information, verifying its correctness or deducting new information.

Rules are included in the ontology for the inference of new knowledge in the ontology from the knowledge already included in it. Their chaining mechanism is not explicitly declared, although WebODE's inference engine uses backward chaining.

Procedures are used for declaring sequences of actions. Currently, the user is free to use any syntax for these components, because it is too much tight to the target language in which the ontology will be used.

The axiom generator, which will be described later in this paper, allows the user create axioms more easily than if they were created from scratch. WebODE also provides a library of axiom patterns for common used expressions.

3.7. Instances

There are two kinds of instances that can be created in WebODE: instances of concepts and instances of relations (also called facts).

Instances of concepts represent elements of a given concept. They have their own **name**, and a set of **instance attributes** with their **values**. Instance attributes are inherited from the concept they belong to and its superclasses.

Instances of relations are used to represent a relation that holds between individuals (instances of concepts) in the ontology. They have their own **name**, the names of the **relation** and the **instances** that participate in it.

WebODE allows grouping both kinds of instances in sets of instances. **Instance sets**, which are described by their **name** and an optional **description**, allow the distributed use of the ontology in different frameworks. In other words, the same ontology can be instantiated for different applications, and instances in an instance set are independent from instances in another one. This, along with the import/export features, permits the isolation of the main two parts of an ontology: its conceptualization and its instances (the knowledge base).

3.8. References

References are used for adding bibliographic references in the ontology. The information needed for references is their **name** and an optional **description**. They can be attached to any component of the WebODE's knowledge model.

3.9. Properties

They are used to describe algebraic properties of ad-hoc relations. They are divided in two groups:

- **Built-in properties**: reflexive, irreflexive, symmetric, asymmetric, antisymmetric and transitive.
- **Ad-hoc properties**. The user can define them and attach them to ad-hoc binary relations to describe either algebraic or other kinds of properties of them.

3.10. Imported terms

Imported terms are components that are included in the current ontology from other ontologies. The user must provide their **name**, the **host** for retrieving the term from, the name of the **ontology** where to retrieve the term from and the **original term name**.

Currently, only concepts from other ontologies can be imported into WebODE. In the future, this will be expanded to any kind of components of the ontology (groups, relations, axioms, etc.).

4. WEBODE ARCHITECTURE

The architecture of the WebODE workbench is explained in this section, according to the classical three tiers architecture commonly found in web applications: data tier, business logic tier and presentation tier.

4.1. Data Tier

Ontologies are the central element in our workbench. They can be stored in a relational database with JDBC support (it has been tested both in Oracle and MySQL).

The module for database access is included as a core service inside the Minerva Application Server (which is explained later in this section). Its main features are the optimization of connections to the database (*connection pooling*) and transparent *fault tolerance* capabilities.

4.2. Business Logic Tier

This tier usually is divided in two different ones: the presentation sub-tier and the logic sub-tier.

The **presentation sub-tier** is responsible for generating the content to be presented in the user's browser. It also handles user requests from the client (form handling, queries, etc.) and forward them to business logic services. *Servlets and*r JSPs (*Java Server Pages*) are used in it.

The **logic sub-tier** comprises the applications' business-logic services. All the implemented services are available from the Minerva Application Server, through RMI-IIOP technology. We distinguish two groups of services: services from the Minerva Application Server, which are not tied specifically to the WebODE workbench but can be used by any other service, and business-logic services for WebODE, which are specific to this workbench.

Modules from Minerva Application Server

This application server has been developed in our lab. In this subsection we will describe its main modules:

Authentication module. All the authentication and security controls in the application server are based on this module. It allows managing access control lists for all the services of applications built upon the server, groups for sharing ontologies, information protection, etc.

Currently, it uses an internal format for storing and accessing information. However, it is possible to develop additional modules for the authentication of users using other authentication systems (from Windows NT, UNIX, LDAP, etc.). This would allow the integration of the workbench in the authentication schema of the organization.

Log module. This module is in charge of auditing tasks. Its verbose level can be configured, depending on the audit needs for the system.

Administration module. It allows the administration of the application server by using the *Minerva Management Console (MMC*), which allows the server administrator to manage locally or remotely every installed service, to start and stop services, to manage users, groups and access control lists through the authenticator service, etc.

Thread management module. This module optimizes the use of threads in the server for any task, using thread-pooling techniques, which improve drastically their execution time. Additionally, it is possible to change thread priorities: some tasks can be executed before other ones.

Planning module. This module, which depends on the thread management module, allows the planning of periodical tasks, such as cache management, periodical backups, ontology consistency checking, etc.

Backup module. Using this module, ontologies can be safely stored in any destination (which is configurable). Backups can be scheduled as needed.

This service makes use of the planning and the ontology XML exportation services. This last service will be explained later in this section.

Business logic modules for WebODE workbench

These modules provide services for the WebODE ontology editor, although they can be used for any other application.

Ontology access service. This module is in charge of managing the ontologies' conceptual model, by inserting, deleting and modifying the definitions of all the terms in a domain.

It uses the database access service, and, optionally, cache and consistency check services, which are explained below.

Cache module. This module, which uses the database access and planning services, speeds up the access to ontologies, using several caching techniques that increase the performance of the ontology access service.

Consistency check module. This module also uses the database access and planning services from the Minerva application server. It performs consistency checks during taxonomy building, as presented in [15], decoupling these verifications from the ontology access service.

Ontology access API. Ontologies can be accessed from other applications through this well-defined API. This API is supported by the Minerva application server and can be accessed through RMI-IIOP.

XML ontology exportation module. It exports ontologies to valid XML, according to a well-defined DTD. This XML code can be used by other applications able to use this format or for later importations of the ontologies into other instances of the WebODE workbench.

XML ontology importation module. It imports ontologies in the XML format described by the DTD used in the XML exportation service. These ontologies must also accomplish consistency rules used by the consistency check service.

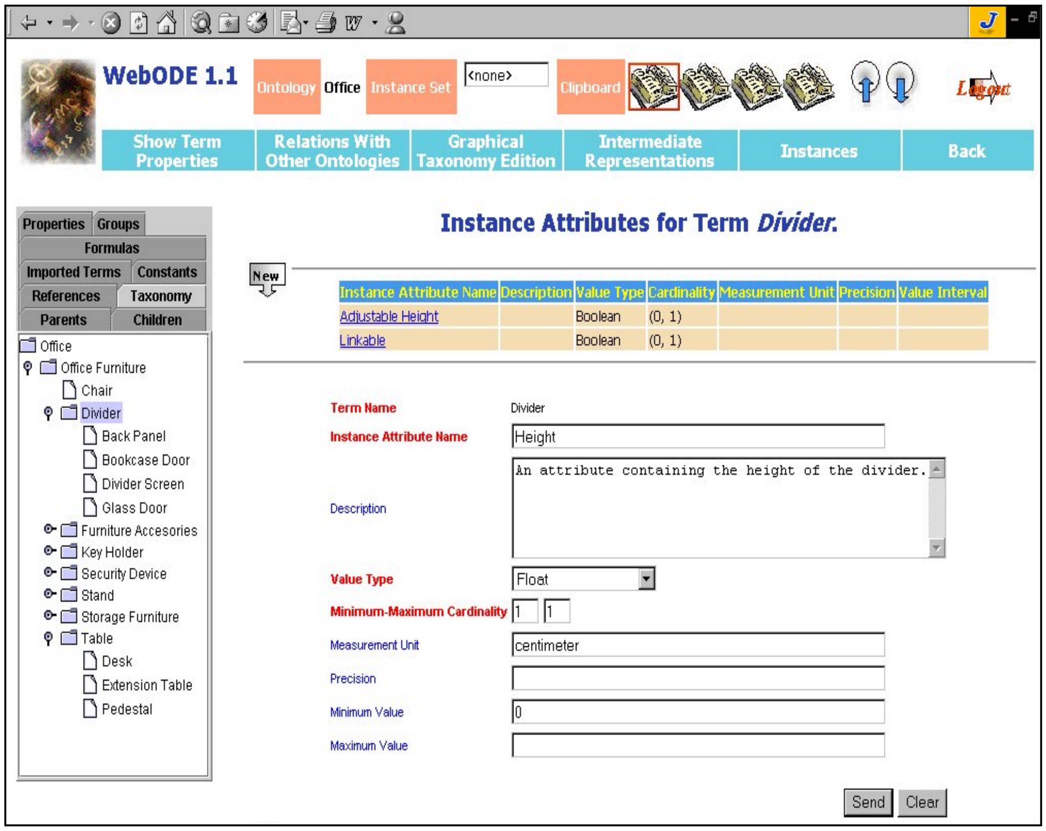
Ontology languages exportation/importation. Currently, several services exist in WebODE for the exportation and importation of ontologies to the following languages: RDF(S), OIL, DAML+OIL, X-CARIN and FLogic.

OKBC-based inference engine. It allows the ontology developer to perform queries and inferences on the ontology. The user can use predefined access primitives, which are based in the OKBC protocol, and create his/her own Prolog programs to perform inferences reusing the primitives already provided. It is based on Ciao Prolog.

Axiom prover module. It makes use of the inference engine, allowing the ontology developer to test if knowledge currently included in the ontology is consistent with its axioms. Each axiom is translated into Horn clauses and can be tested independently from the other ones.

Documentation module. WebODE ontologies are automatically documented in different formats, such as HTML tables (intermediate representations of METHONTOLOGY), HTML documents or XML files. The whole ontology or parts of it can be selected for this documentation generation. Views generated with OntoDesigner can be also selected for their documentation.

WebPicker: Ontology Acquisition from Web Resources. WebPicker is a service for the ontology acquisition from web resources that has been used for the acquisition of several standards and initiatives of products and services classifications in the e-commerce domain (UNSPSC, e-cl@ss and RosettaNet) as described in detail in [6].



Instance Attributes for Term *Divider*.

Instance Attribute Name	Description	Value Type	Cardinality	Measurement Unit	Precision	Value Interval
Adjustable Height		Boolean	(0, 1)			
Linkable		Boolean	(0, 1)			

Term Name	Divider
Instance Attribute Name	Height
Description	An attribute containing the height of the divider.
Value Type	Float
Minimum-Maximum Cardinality	1 1
Measurement Unit	centimeter
Precision	
Minimum Value	0
Maximum Value	

Figure 2. Snapshot of WebODE's ontology editor while editing an instance attribute of a concept.

Information represented in web resources is transformed into a conceptual model specified in the XML syntax of WebODE, which is imported later into WebODE, so that its ontology editor can be used to redesign it.

OntoMerge: Ontology Merge. This service performs the merge of concepts (and their attributes) and relations of two ontologies built for the same domain. First, it assists the revision of both ontologies, based on a set of design criteria and semantic and syntactic relationships among the components of the ontology. Later, it uses natural language resources for establishing relationships between both ontologies. It performs a supervised merge of components from both ontologies using this information. Finally, it assists the final revision of the resulting merged ontology.

OntoCatalogue: Catalogue Generation from Ontologies. This service generates electronic catalogues out from ontologies, taking into account several configuration parameters, such as the depth of the taxonomy of products, attributes to be generated, the mappings between relations in the ontology and links in the catalogue, navigation hints through the catalogue, parts of the taxonomy to be generated, etc.

The catalogue generation from ontologies ensures a good and rich classification of products/services in it.

4.2.1. User Interface Tier: WebODE Ontology Editor

The WebODE ontology editor is an application for the development of ontologies at the knowledge level, based on the knowledge model already presented, which uses most of the services that have been presented above. Its user interface uses HTML, CSS (*Cascading Style Sheets*) and XML (*Extended Mark-up Language*). JavaScript and Java are used for several kinds of user validations.

Some specialized applets have been also included in the user interface, such as OntoDesigner, the axiom manager, the ontology browser and the clipboard.

The design rationale for this user interface is based on an easy-to-use and clarity basis. Figure 2 shows one of the screens of the editor, while including a new instance attribute for a concept. We will explain the most relevant components in this figure:

The clipboard applet is available in the upper part of the screen. It is used to copy and paste components' definitions, which is useful when creating components that are very similar to others. It has enough space for four definitions.

The ontology browser is placed on the left. It aids the navigation through the taxonomy of concepts, formulae, references and imported terms in the ontology. New components can be added by just double clicking on it and

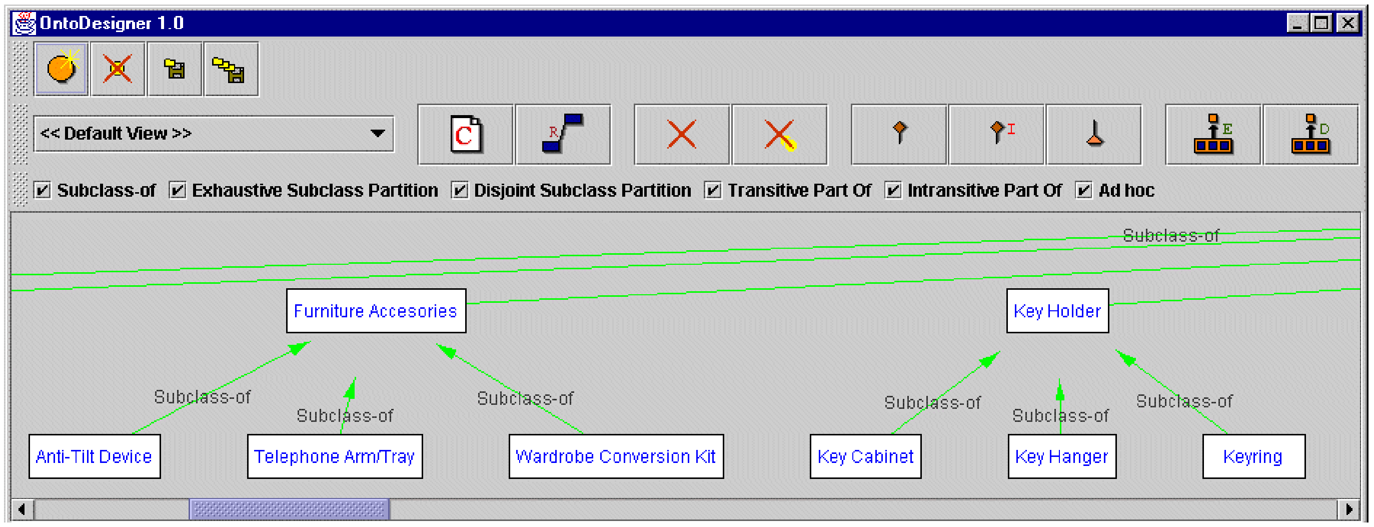


Figure 3. OntoDesigner.

filling the form that appears in the middle of the screen, and contextual menus arise when right-clicking on any of the visualized components.

This user interface also includes the functionalities of exporting/importing ontologies into XML or varied ontology languages, inference engine and documentation.

OntoDesigner. OntoDesigner is a graphical user interface for the visual construction of taxonomies of concepts and ad-hoc relations between concepts, which is integrated in the WebODE ontology editor as an applet. Figure 3 shows a snapshot of OntoDesigner while editing an ontology on the domain of office furniture.

Using OntoDesigner, the user can create different views of the edited ontology, so that the visualization of parts of the ontology can be customized while creating it. Moreover, the user can decide at any time whether showing or hiding different kinds of relations (either predefined or ad-hoc) between concepts, in the sense of a graphical prune.

Axiom Manager. This applet is used to ease the management of formulae in the WebODE ontology editor. It allows the user to create axioms using a graphical interface and provides functionalities such as an axiom library, axiom patterns and axiom parsing and verification.

5. RELATED WORK

WebODE has a strong relationship with ODE [3]. Both applications allow building ontologies at the knowledge level, and translators are used to implement them in different ontology languages. ODE was created as a classical application for single users and was difficult to extend. Furthermore, ontologies were stored in a Microsoft Access database, which proved to be inefficient when dealing with large ontologies. However, while ODE knowledge model is flexible, WebODE knowledge model is fixed, as has been explained in this paper.

Protégé2000 and OntoEdit are ontology development tools developed at the same time than WebODE, and using a similar design rationale, although they are not web-based but stand-alone applications. In fact, they share many functionalities (ontology edition, ontology documentation, ontology exportation and importation into XML and other languages). Moreover, Protégé2000 has been developed using a plug-in architecture, where new services can be added easily to the environment. However, WebODE integrates all its services in a well-defined architecture, stores its ontologies in a relational database (avoiding the use of text files) and provides additional services such as the inference engine, the axiom builder, ontology acquisition or catalogue generation.

OilEd was developed in the context of the OntoKnowledge [22] EU project for the easy development of OIL ontologies. It is not intended as a complete ontology editor, but just “the Notepad for OIL ontologies”.

Other “classic” editors, such as WebOnto, Ontolingua and OntoSaurus, can be used for the edition of ontologies in a specific language (OCML, Ontolingua and LOOM, respectively). They do not use databases for storing ontologies.

6. CONCLUSIONS

In this paper, we have stated the need for a workbench for ontological engineering that allows:

- the development and management of ontologies,
- a wide use and integration of ontologies using a set of useful ontology middleware services, and
- the rapid development of ontology-based applications for their integration in enterprise information systems.

We have presented the WebODE workbench as a solution for this needs, describing its expressive knowledge model for representing ontologies, several built-in services and additional reusable services, such as WebPicker, OntoMerge and OntoCatalogue.

Its ontology editor integrates in a common user interface most of the activities of the ontology life cycle, using the services available in the workbench. Its most interesting functionalities are: multiple-users support, guided conceptualization through the use of a very intuitive and simple user interface, multiple choice clipboard for easily copying and pasting components, complete consistency checks to ensure that the ontology contains valid knowledge, easy taxonomy edition either by using the form based user interface or a more complex and powerful graphical editor (OntoDesigner), an advanced term import providing by reference and by value fashions, instance handling independent from the ontology conceptualization, an API for accessing ontologies from any application using RMI or CORBA, and, finally, maximum interoperability thanks to the use of XML and several ontology specification languages.

This workbench has been successfully used in several projects: B2B and B2C ontology creation and reengineering in MKBEEM (IST 1999-10589), ontology acquisition through Webpicker in ContentWeb (*UNSPSC*, *RosettaNet* and *e-cl@ss*), ontology building and ontology metrics in (Onto)^{2}Agent (*Reference Ontology*), ontology building in project UPM:AM-9819 "Environment Ontology" (*Elements* and *Environmental Ions*) and electronic catalogues merging in MRO.

In the future we will provide extra services both to the WebODE ontology editor and the middleware area, such as ontology translation manager, ontology configuration management capabilities, ontology upgrading, etc.

ACKNOWLEDGEMENTS

This work is supported by a FPI grant funded by UPM and by the project ContentWeb funded by MEC. It would not have been possible without the help of J.P. Pérez, O. Vicente, J. Ramos, R. de Diego, A. López, V. López and E. Mohedano, in the implementation and/or tests of WebODE, and developers of ODE (M. Blázquez and J.M. García).

REFERENCES

1. Benjamins, V.R., Fensel, D., Decker, S., Gómez-Pérez, A. *(KA)^2: Building Ontologies for the Internet: a Mid Term Report*. IJHCS, 51:687-712. 1999.
2. Berners-Lee, T., Fischetti, M. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. Harper. S Francisco. 1999.
3. Blázquez, M.; Fernández-López, M.; García-Pinar, J.M.; Gómez-Pérez, A. *Building Ontologies at the Knowledge Level using the Ontology Design Environment*. KAW98. Banff, Canada. 1998.
4. Chalupsky, H. *OntoMorph: A Translation System for Symbolic Knowledge*. KR-2000. 471-482. 2000.
5. Chaudhri V. K.; Farquhar A.; Fikes R.; Karp P. D.; Rice J. P. *The Generic Frame Protocol 2.0*. Technical Report, Stanford University.1997.
6. Corcho, O., Gómez-Pérez, A. *WebPicker: Knowledge Extraction from Web Resources*. NLDB'01. Madrid. June, 2001.
7. DAML+OIL. http://www.daml.org
8. Domingue, J. *Tadzebao and Webonto: Discussing, Browsing and Editing Ontologies on the Web*. KAW98. Banff, Canada. 1998.
9. Domingue, J., Motta, E. *A Knowledge-Based News Server Supporting Ontology-Driven Story Enrichment and Knowledge Retrieval*. EKAW 1999.
10. Duineveld, A.; Studer, R.; Weiden, M; Kenepa, B.; Benjamis, R. *WonderTools? A comparative study of ontological engineering tools*. KAW99. Banff. 1999.
11. Farquhar A., Fikes R., Rice J., *The Ontolingua Server: A Tool for Collaborative Ontology Construction*. 10th Knowledge Acquisition for Knowledge-Based Systems Workshop, Banff, Canada. 1996.
12. Fensel, D., Angele, J., Decker, S., Erdmann, M., Schnurr, H., Staab, S., Studer, R., Witt, A. *On2broker: Semantic-Based Access to Information Sources at the WWW*. WebNet 99. Honolulu. USA. October, 1999.
13. Fernández, M.; Gómez-Pérez, A.; Pazos, J.; Pazos, A. *Building a Chemical Ontology using methontology and the Ontology Design Environment*. IEEE Intelligent Systems and their applications. #4 (1):37-45. 1999.
14. Fridman, N., Musen, M. *PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment*. AAAI-2000. Austin, Texas. August, 2000.
15. Gómez-Pérez, A. *Evaluation of Ontologies*. International Journal of Intelligent Systems. 16(3). March, 2001.
16. Horrocks, I., Fensel, D., Harmelen, F., Decker, S., Erdmann, M, Klein, M. *OIL in a Nutshell*. EKAW'00. Juan les Pins. France. October, 2000.
17. Kifer, M.; Lausen, G.; Wu, J. *Logical Foundations of Object-Oriented and Frame-Based Languages*. Journal of the ACM. 1995.
18. McGuinness, D., Fikes, R., Rice, J., Wilder, S. *The Chimaera Ontology Environment*. AAAI-2000. Austin, Texas. August, 2000.
19. MKBEEM. http://mkbeem.elibel.tm.fr
20. *OILEd*. http://img.cs.man.ac.uk/oil/
21. *OntoEdit*. http://www.ontoprise.de/co_produ_tool3.htm
22. *OntoKnowledge*. http://www.ontoknowledge.org
23. RDF. http://www.w3.org/TR/REC-rdf-syntax/
24. Swartout, B.; Ramesh P.; Knight, K.; Russ, T. *Toward Distributed Use of Large-Scale Ontologies*. AAAI Symposium on Ontological Engineering. Stanford. USA. March, 1997.
25. *Using Protégé-2000 to Edit RDF*. Technical Report. Stanford University. http://www.smi.Stanford.edu/ projects/protege/protege-rdf/protege-rdf.html.