AI Magazine Volume 24 Number 3 (2003) (© AAAI)

WebODE in a Nutshell

Julio César Arpírez, Oscar Corcho, Mariano Fernández-López, and Asunción Gómez-Pérez

■ WebODE is a scalable workbench for ontological engineering that eases the design, development, and management of ontologies and includes middleware services to aid in the integration of ontologies into real-world applications. WebODE presents a framework to integrate new ontology-based tools and services, where developers only worry about the new logic they want to provide on top of the knowledge stored in their ontologies.

In the last decade, several tools for ontology development have appeared: OILed (Bechhofer et al. 2001), OntoEdit (Sure et al. 2002), ontolingua (Farquhar, Fikes, and Rice 1997), OntoSaurus (Swartout et al. 1997), protégé-2000 (Noy, Fergerson, and Musen 2000), WebODE (Corcho et al. 2002),[1] WebOnto (Domingue 1998), and so on. Comparative studies of some of these tools appeared in Duineveld et al. (1999).[2]

These tools can be used for a great range of activities in the ontology development process, such as ontology design, browsing and implementation, ontology importation, ontology merge and alignment, and ontology-based resource annotation. However, only OntoEdit, protégé-2000, and WebODE give integrated support to most of the activities of the ontology development process, and only OntoEdit and WebODE give support to an ontology-building methodology (OnToKnowledge [Sure and Studer 2002] and methontology [Fernández-López et al. 1999], respectively).

Moreover, most of the tools have usually been built as isolated, independent tools that are incapable of interoperating. Consequently, ontologies owned by different organizations, built using different tools and implemented in different languages, are difficult to exchange between ontology platforms. The need for studies of tools interoperability is being addressed by the Special Interest Group on Enterprise-Standard Ontology Environments of the European Information Society Technologies thematic network OntoWeb (IST-2000-29243).

Taking into account the functions currently provided by available tools and other interesting functions that could be added on top of them, we propose the need for an integrated ontological engineering workbench (Corcho et al. 2002) supporting three main groups of activities (figure 1):

First are ontology development and management activities that for ontology development include ontology edition, browsing, learning, merge, alignment, translation, and evaluation and for ontology management include ontology configuration management and evolution and documentation.

Second are ontology middleware services that allow the easy use and integration of ontology-based technology into existing and future information systems, such as services for ontology library administration and access and ontology upgrading, querying, and metrics.

Third are ontology-based application development suites that provide multiple means for the rapid development and integration of ontology-based applications. Ideally, they will be the last step toward a real integration of ontologies into enterprise information systems.

In the context of this framework, WebODE has been developed as a scalable ontological engineering workbench that gives support to most of the ontology development and management activities presented in figure 1. WebODE also includes middleware services to aid in the integration of ontologies into real-world

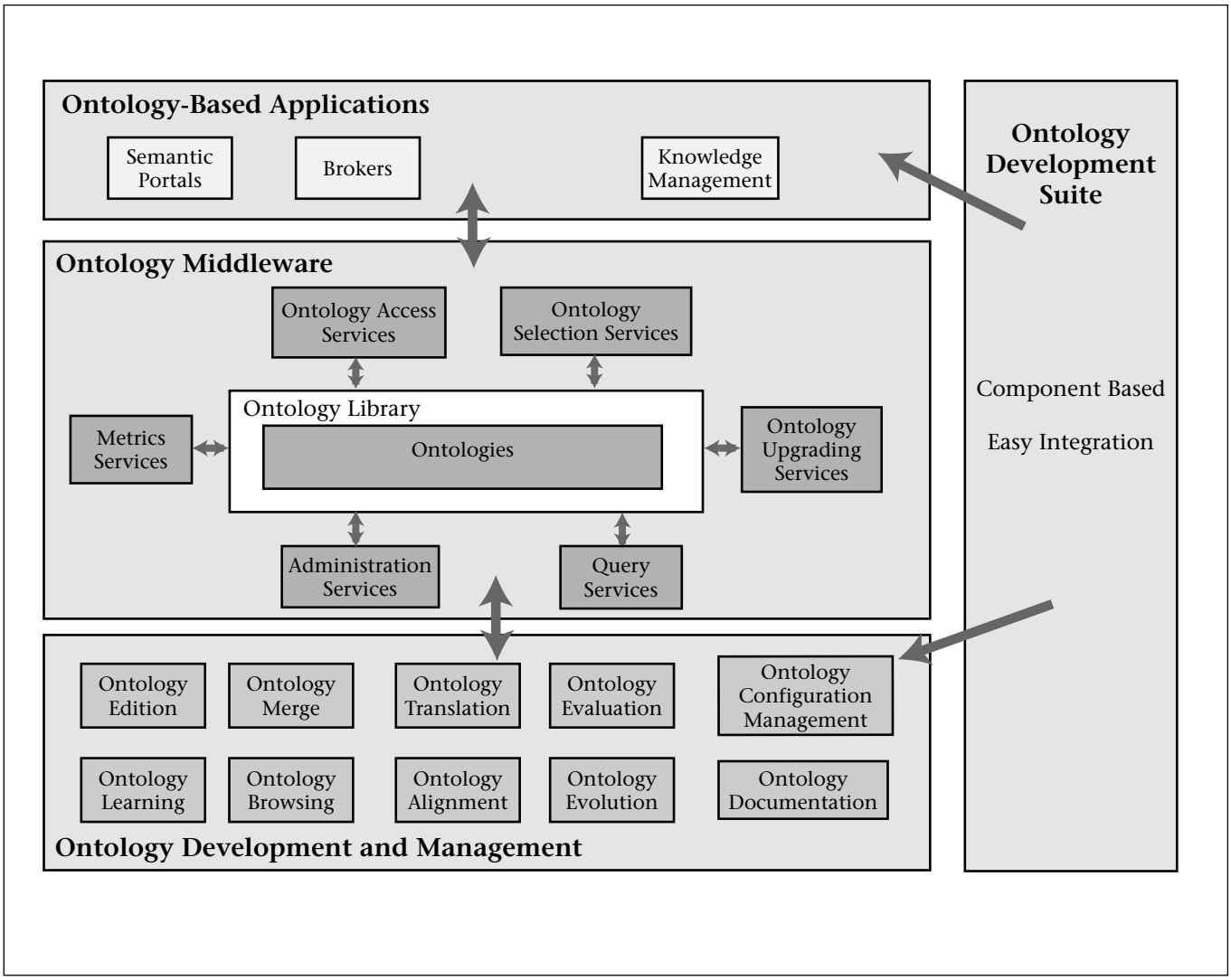


Figure 1. Main Modules of an Ontological Engineering Workbench (adapted from Corcho et al. [2002]).

applications as well as rapid development tools for building ontology-based web portals (ODESEW) and ontology-based knowledge management applications (ODEKM). Finally, WEBODE was created to provide technological support to METHONTOLOGY (Fernández-López et al. 1999), a methodology for ontology construction. Nevertheless, this fact does not prevent it from being used based on other methodologies or no methodological approach at all.

The next sections describe the architecture and knowledge model of the WEBODE workbench. Then we present the most important features of the WEBODE ontology editor, and we describe how to use WEBODE ontologies in ontology-based applications.

WEBODE Architecture

WEBODE is built completely with JAVA technology, and it is designed according to an *n*-tier architecture, partitioned into three independent layers: (1) front ends (presentation), (2) middle (business logic), and (3) back ends (database management).

Current WEBODE front ends, such as the ontology editor, ODEKM, and ODESEW, are implemented with HTML and JAVA applets. Because the communication between clients and the middle layer is carried out with HTTP, front ends can also use web service technologies such as SOAP and WSDL.[3,4]

The middle tier is based on an application server, which makes it easy to create and add new services. The application server improves

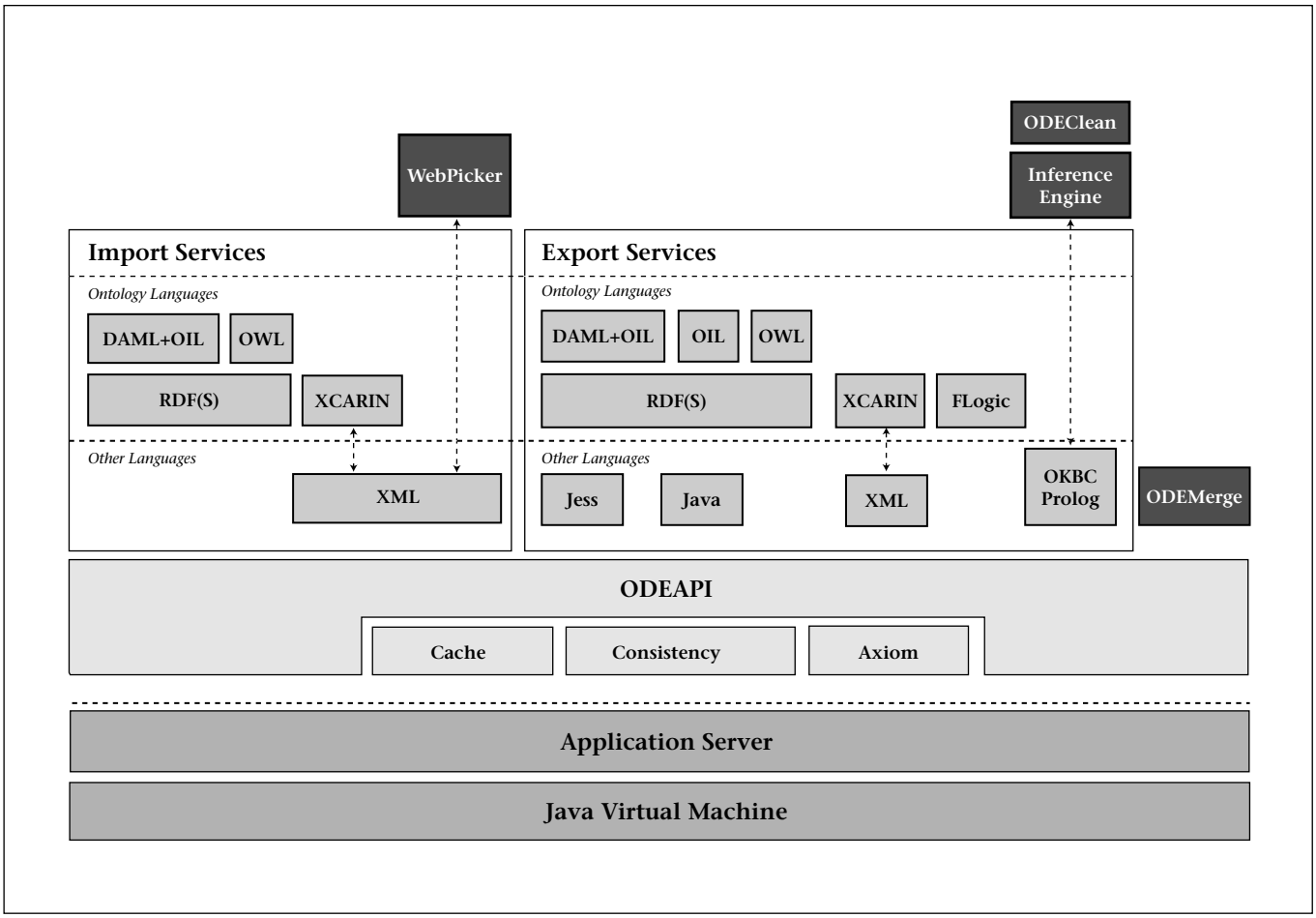


Figure 2. Architecture of WebODE Services.

WebODE's flexibility, scalability, and integration with third-party solutions and includes basic functions of authentication, log, administration, thread management, scheduling, backup, and database access. WebODE's current services are shown in figure 2. The most important service is the WebODE ontology access application programming interface (API) (ODE API), which allows accessing ontologies in the workbench. As shown in the figure, the ODE service relies on other services for ontology caching (improving access time to ontology components), consistency checking, and axiom checking. A wide range of services for importing and exporting ontologies are available, as we describe in the next section, and some other services rely on them. For example, WebPicker wraps web resources and converts them into ontologies and relies on the XML import service. WebODE's inference engine uses ontologies exported in PROLOG, and ODEClean uses the WebODE's inference engine. Another service available is ODEMerge, which merges ontologies.

WebODE ontologies can be stored in any relational database with JAVA database connectivity (JDBC) support (WebODE has been tested with ORACLE and MySQL). This back end optimizes database connections by connection pooling and provides transparent fault-tolerance capabilities. In addition, its physical model is tuned for maximum performance.

WebODE's Knowledge Model

Ontologies in WebODE are conceptualized with a very expressive knowledge model (Corcho et al. 2002), which is based on the reference set of intermediate representations of METHONTOLOGY (Fernández-López et al. 1999). The following ontology components are included in WebODE's knowledge model: concepts and their attributes (both instance and class attributes); concept groups, which represent sets of disjoint concepts; concept taxonomies and disjoint and exhaustive class decompositions; ad hoc binary relations between concepts, which can be characterized by rela-

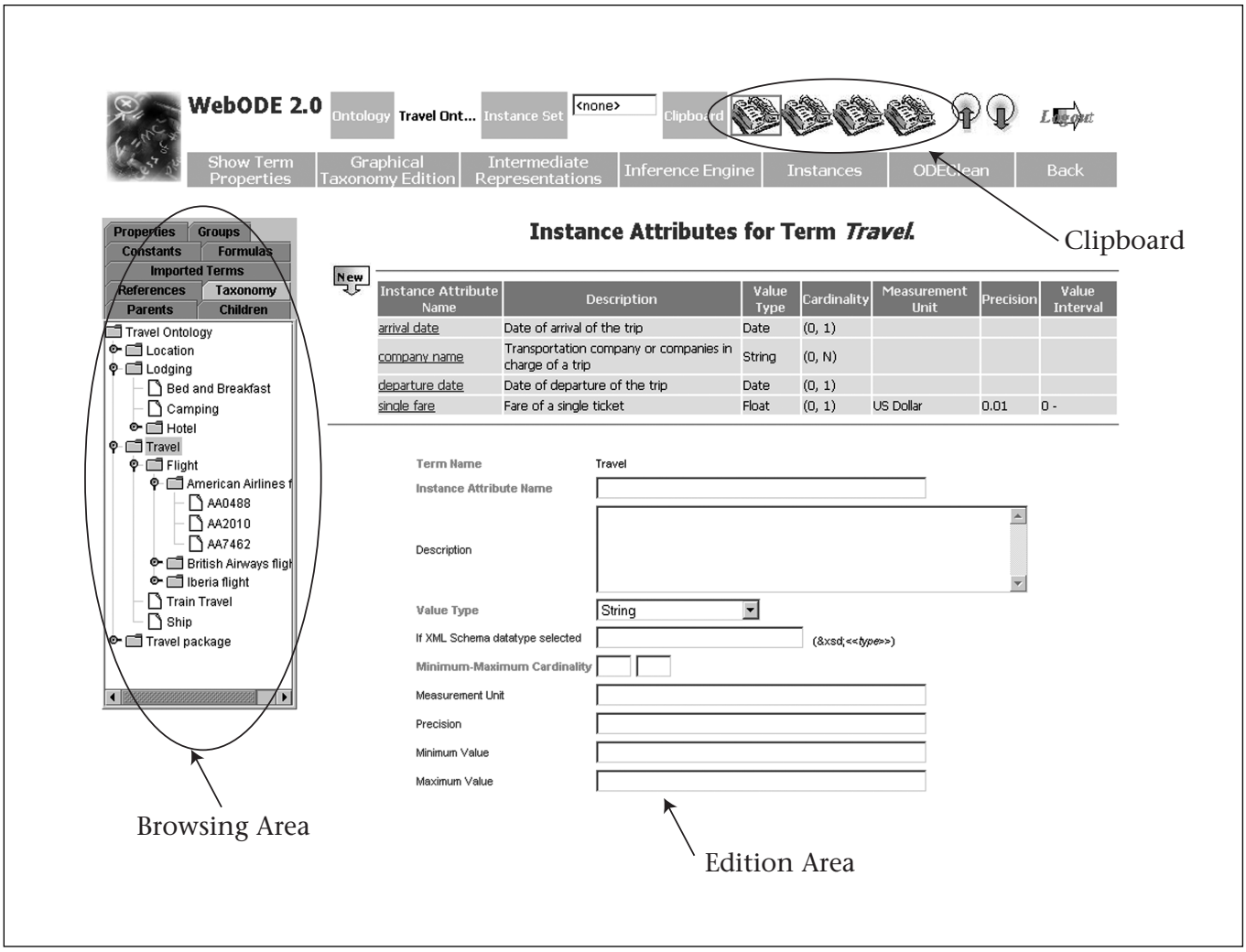


Figure 3. WebODE Ontology Editor.

tion properties (symmetry, transitiveness, and so on); constants; formal axioms, expressed in first-order logic; rules; and instances of concepts and relations. Bibliographic references, synonyms, and acronyms can be attached to any of the aforementioned ontology components.

The WebODE knowledge model also allows referring to terms from other ontologies (imported terms), which are identified by means of uniform resource identifiers (URIs).

WebODE instances are defined inside instance sets, which allows creating different instantiations for the same ontology that are independent from each other. For example, if we have created an ontology about books and libraries, we can instantiate it in different instance sets, one for each library that is using the ontology.

WebODE Ontology Editor

The WebODE ontology editor is a web application that allows the collaborative construction of ontologies at the knowledge level. In this section, we present WebODE's ontology edition functions, its ontology documentation options, its import and export facilities, its consistency-checking functions, its inference engine, and its ontology evaluation functions.

Ontology Edition

The user interface of the WebODE ontology editor is divided into three main parts: (1) HTML forms to edit all ontology components except formal axioms and rules; (2) OntoDesigner to graphically edit concepts, relations, and concept taxonomies; and (3) the WebODE axiom builder (WAB) to edit formal axioms and rules. Although the HTML user interface and OntoDe-

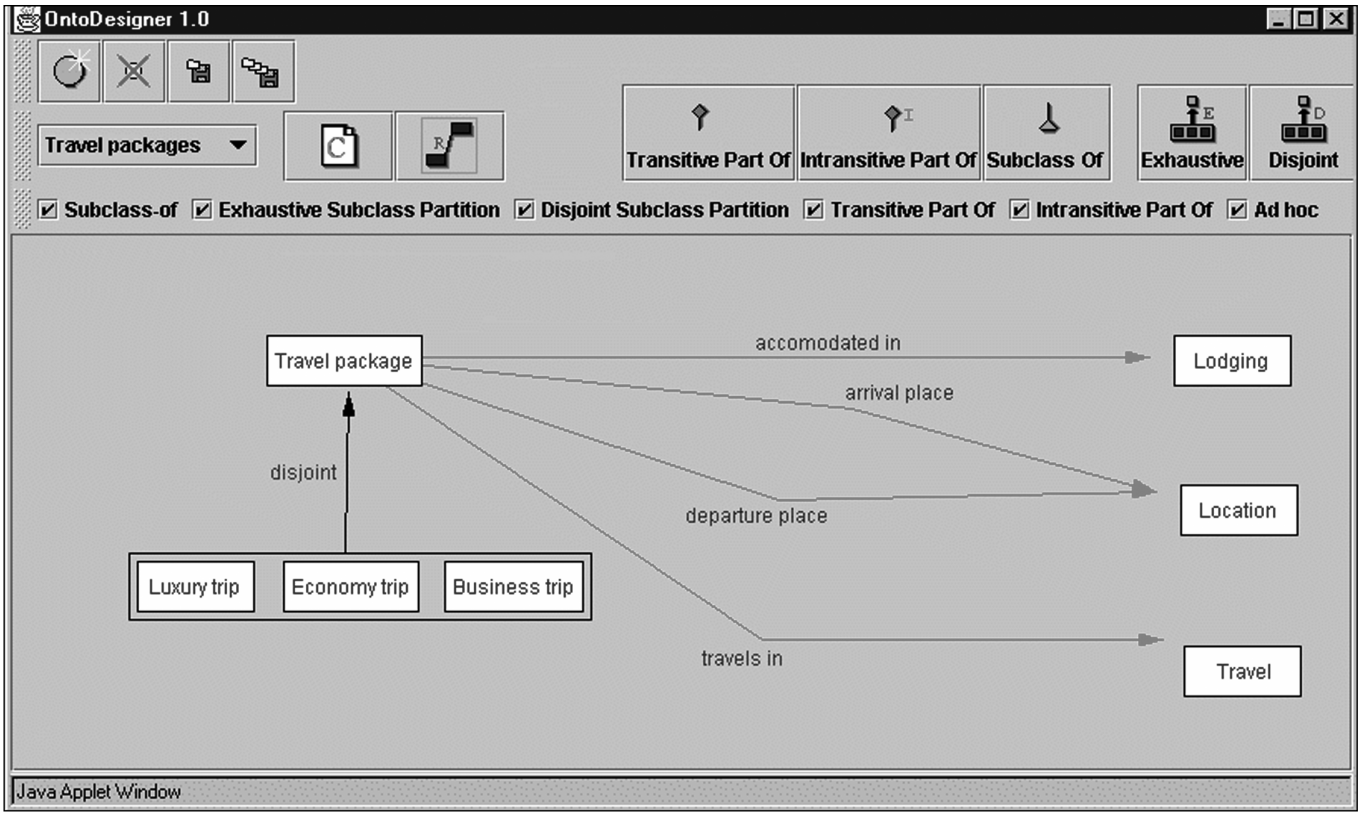


Figure 4. Screenshot of ONTODESIGNER.

SIGNER allow creating lightweight ontologies, WAB allows adding axioms and rules to convert them into heavyweight ontologies (Studer, Benjamins, and Fensel 1998).

Figure 3 shows a screenshot of the HTML interface. It is creating an instance attribute of the concept *travel*. There are three main components of this user interface: (1) browsing area, (2) clipboard, and (3) edition area.

The *browsing area* is used to navigate the whole ontology, including operations to create new terms and modify or delete existing terms.

The *clipboard* is used to easily copy and paste information between forms.

The *edition area* presents forms to insert, delete, and update ontology terms (concept, attribute, relation, and so on) and tables with existing ones. For example, figure 3 shows four attributes of the concept *travel* that have already been attached to it—(1) arrival date, (2) company name, (3) departure date, and (4) single fare—and a form to create a new attribute for this concept.

ONTODESIGNER is a graphic user interface (GUI) used to build concept taxonomies with the following set of predefined relations: subclass-of, disjoint decompositions and exhaustive partitions, and transitive and nontransitive part-of relationships. Ad hoc relations between concepts can also be created with this interface. Figure 4 shows a screenshot of ONTODESIGNER editing an ontology on the domain of traveling.

With ONTODESIGNER, users can create different views of an ontology to highlight or customize distinct parts of its concept taxonomy, relations between concepts, and so on. Moreover, users can decide at any time whether to show or hide different kinds of relations (either predefined or ad hoc), in the sense of a graphic prune. This feature helps in the manual evaluation of the relations contained in an ontology.

WAB is a graphic editor that allows creating first-order logic axioms and rules. Figure 5 shows an axiom in WAB that states that "trains that depart from Europe must arrive in Europe." The buttons below the text box help with writing logical expressions with quantifiers and logical connectives, and the dropdown lists below them allow the easy inclusion of ontology concepts, their attributes and ad hoc relations, and constants.

Once an axiom is completely written in the text box, it is parsed and transformed into Horn clauses through a skolemization process.

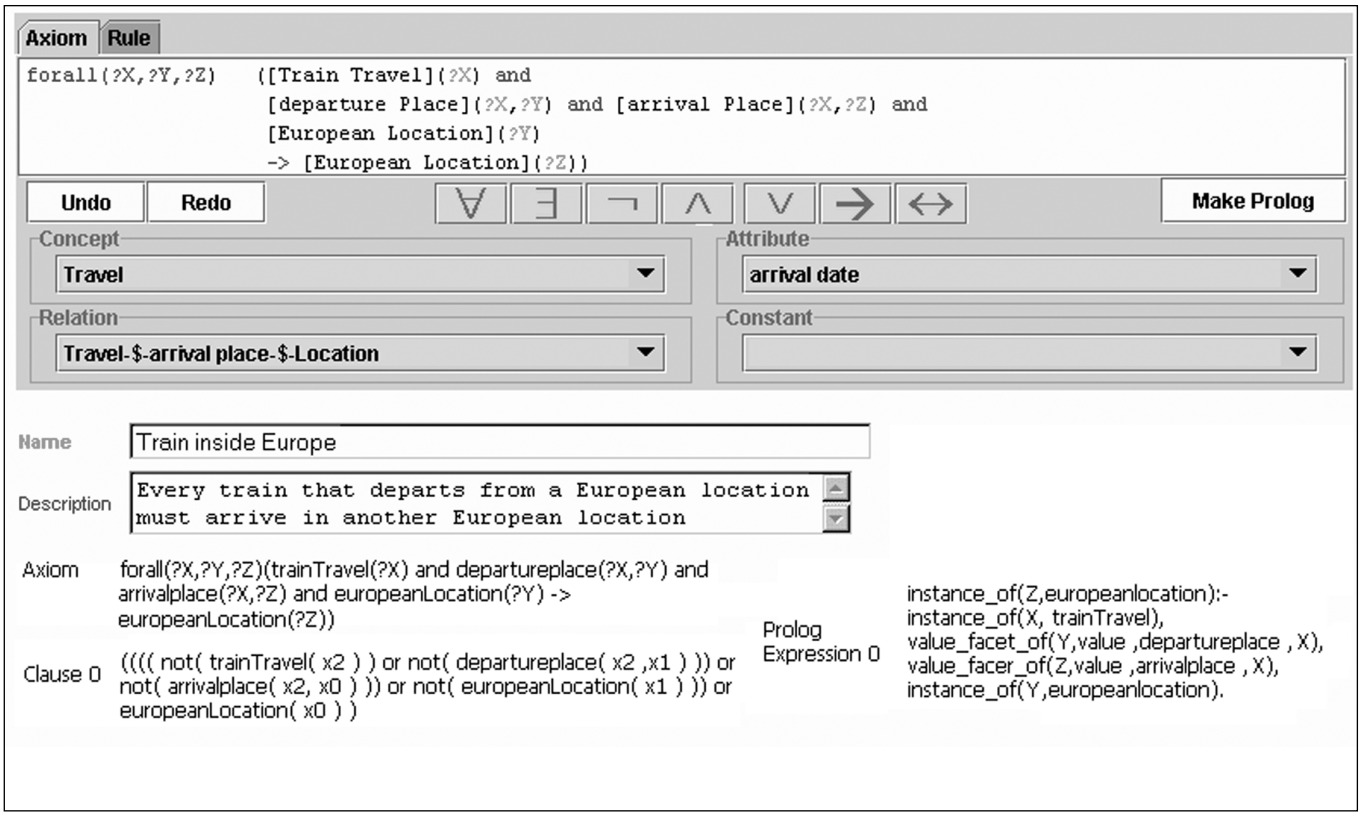


Figure 5. Axiom Edition with WAB.

If it is not possible to transform the axiom into Horn clauses, WAB warns the user. These Horn clauses are then transformed into PROLOG using primitives defined in the open knowledge base connectivity (OKBC) protocol (Chaudri et al. 1997) so that they can be used by the PROLOG inference engine attached to WEBODE.

Rules are created similarly. They are composed of an antecedent and a consequent, and concepts, attributes, relations, and constants can easily be included in the rule by means of drop-down lists. Figure 6 shows WAB creating a rule that states that "trips by ship that depart from Europe are handled by the company CostaCruises." As with formal axioms, rules are also transformed into Horn clauses and then to OKBC-based PROLOG rules.

Ontology Documentation

WEBODE ontologies are automatically documented in different formats, such as the already mentioned METHONTOLOGY's intermediate representations, HTML, and XML. The whole ontology, or parts of it (specific views or instance sets), can be selected for this documentation.

Figure 7 presents the concept dictionary of an ontology in the traveling domain, including all its concepts, their class and instance attributes, instances, and ad hoc binary relations. It also presents a sample result of the HTML documentation service, showing the ontology concepts, the concept taxonomy (the subclass-of relationship between concepts is represented by indentation), the concept attributes, and ad hoc binary relations (with their name and their destination concept).

Ontology Import and Export Services

The WEBODE ontology editor provides translation services for the following ontology languages—XCARIN, FLOGIC, RDF(S), OIL, DAML + OIL, and OWL—and translation services from XCARIN, RDF(S), DAML + OIL, and OWL. Taking into account that the WebODE knowledge model is very expressive, we are able to provide high-quality translations that preserve most of the original information contained in the ontology and take advantage of most of the modeling characteristics of the target languages.

WEBODE ontologies can also be exported and imported from XML, following a well-defined document-type definition (DTD) that uses the same knowledge representation vocabulary that is used to express its knowledge model.

It can also export ontologies into other languages that are not specifically designed for im-

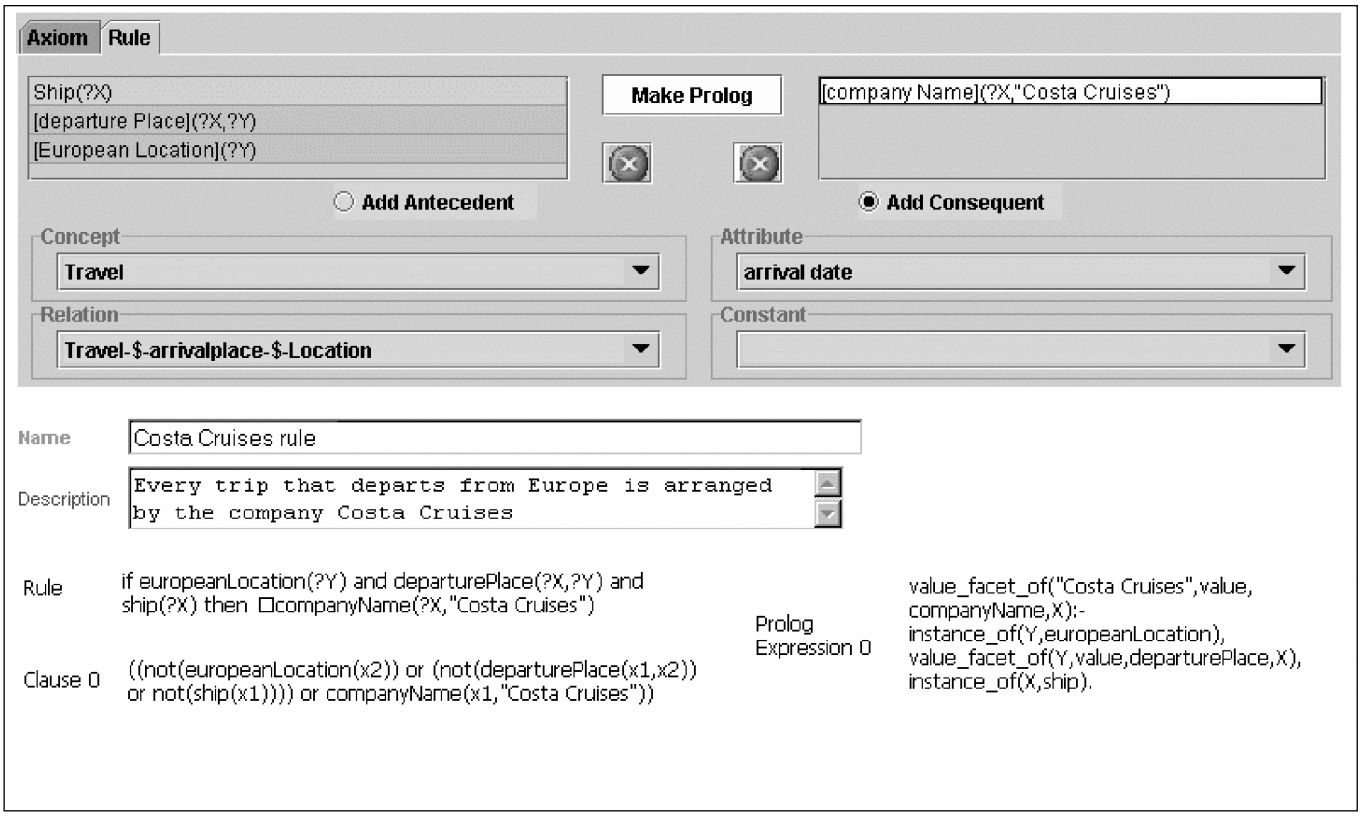


Figure 6. Rule Edition with WAB.

plementing ontologies, such as PROLOG, JAVA, and JESS.

Ontology Consistency Checking

When building ontologies with the WEBODE ontology editor, their consistency is checked. The ontology editor checks type constraints, numeric value constraints, cardinality constraints, and concept taxonomy consistency (that is, common instances of disjoint concepts, loops in the concept taxonomy, and so on). These consistency-checking functions are invoked either from the HTML interface or ONTODESIGNER.

Built-In Inference Engine

The WEBODE ontology editor includes a built-in OKBC-based inference engine, which is based on a subset of the primitives identified in that protocol (Chaudhri et al. 1997). This inference engine is implemented in CIAO PROLOG and can be used to query information about the ontologies and detect more types of inconsistencies with the axioms created with WAB.[5]

Ontology Evaluation

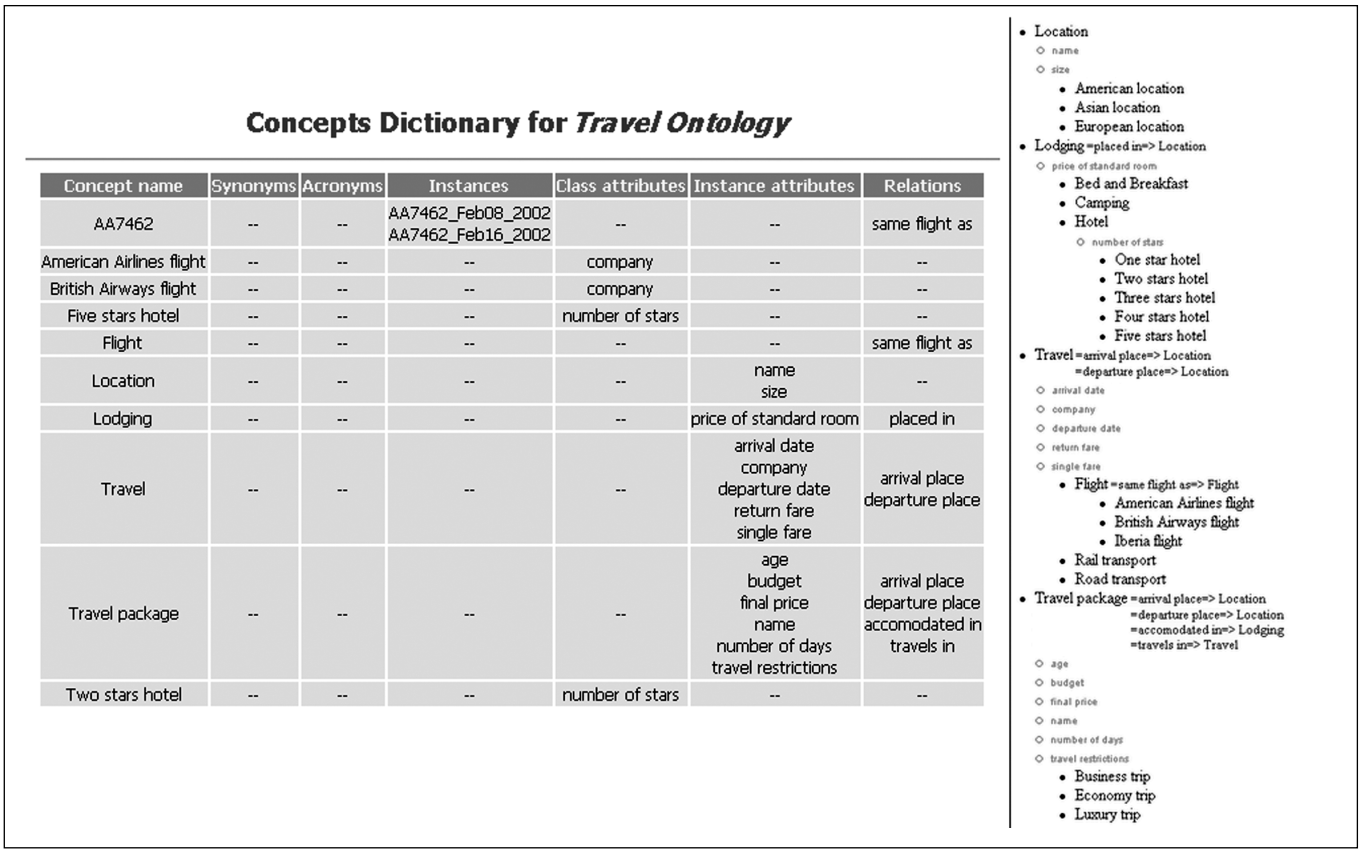
The WEBODE ontology editor gives support to the ONTOCLEAN method (Guarino and Welty 2002) by means of the ODECLEAN service (Fernández-López and Gómez-Pérez 2002). This evaluation service uses the WEBODE inference engine and can be switched on and off so that WEBODE users can decide whether to use it to clean their concept taxonomies according to the notions of rigidity, identity, and unity that are used by this method.

Figure 8 shows ONTODESIGNER cleaning the concept taxonomy of a travel ontology with ODECLEAN. The symbols that appear in the lower part of each concept represent that a concept is rigid (+R), is nonrigid (-R), is antirigid (~R), has unity (+U), and so on. The symbols given in parens are typically used in the method. After applying these metaproperties to the concepts, the user can evaluate the concept taxonomy, and the errors in the taxonomy appear in a separate window and are highlighted in the graph.

How to Use WEBODE Ontologies

Ontologies built with the WEBODE ontology editor or imported into the workbench can be used by ontology-based applications in many different ways:

Ontologies can be accessed and modified



Concepts Dictionary for *Travel Ontology*

Concept name	Synonyms	Acronyms	Instances	Class attributes	Instance attributes	Relations
AA7462	--	--	AA7462_Feb08_2002 AA7462_Feb16_2002	--	--	same flight as
American Airlines flight	--	--	--	company	--	--
British Airways flight	--	--	--	company	--	--
Five stars hotel	--	--	--	number of stars	--	--
Flight	--	--	--	--	--	same flight as
Location	--	--	--	--	name size	--
Lodging	--	--	--	--	price of standard room	placed in
Travel	--	--	--	--	arrival date company departure date return fare single fare	arrival place departure place
Travel package	--	--	--	--	age budget final price name number of days travel restrictions	arrival place departure place accomodated in travels in
Two stars hotel	--	--	--	number of stars	--	--

Figure 7. Different Types of Documentation of the WebODE Ontology Editor.

with the WebODE ontology access API (ODE API in figure 2). Ontology-based applications can be built as services on top of the same application server as WebODE or can access WebODE ontologies remotely with remote method invocation (RMI) or web services.

Ontologies can be exported to XML so that the ontology-based application accesses and modifies the XML file obtained. If the ontologies are modified, they can later be imported into the WebODE workbench if needed.

Ontologies can be exported to several ontology languages (XCARIN, FLOGIC, RDF(S), OIL, DAML + OIL, OWL) and other more general languages (JESS), as presented in the previous section. Specific APIs and inference engines developed by third parties for these languages can be used to access, modify, and reason with these ontologies. For example, if an ontology is transformed into RDF(S), there are several parsers and inference engines that could be used, such as JENA, the ICS-FORTH RDFSUITE, SESAME, and REDLAND. With DAML + OIL ontologies, we can use JENA, FACT, RACER, and so on.

Ontologies can be exported to Prolog and used by the WebODE's OKBC-based Prolog inference engine. We can also use them with other Prolog programs, and integrate them into ontology-based applications that use these Prolog programs.

Ontologies can be exported to JAVA. Each ontology concept is transformed into a JAVA class, each attribute is transformed into a variable, each ad hoc relation is transformed into a class association, and so on. This JAVA code can be compiled and used in other JAVA applications.

Conclusions

In this article, we presented the WebODE ontological engineering workbench and its main contributions to the ontology area, which are summarized as follows:

WebODE gives integrated technological support to several activities of the ontology development process, usually supported by other independent tools that do not interoperate. WebODE gives support to ontology edition, ontology translation, ontology merge, ontology evaluation, and ontology documentation.

WebODE ontology developers do not need to worry about building ontologies directly in an ontology language because WebODE ontologies are automatically translated into many

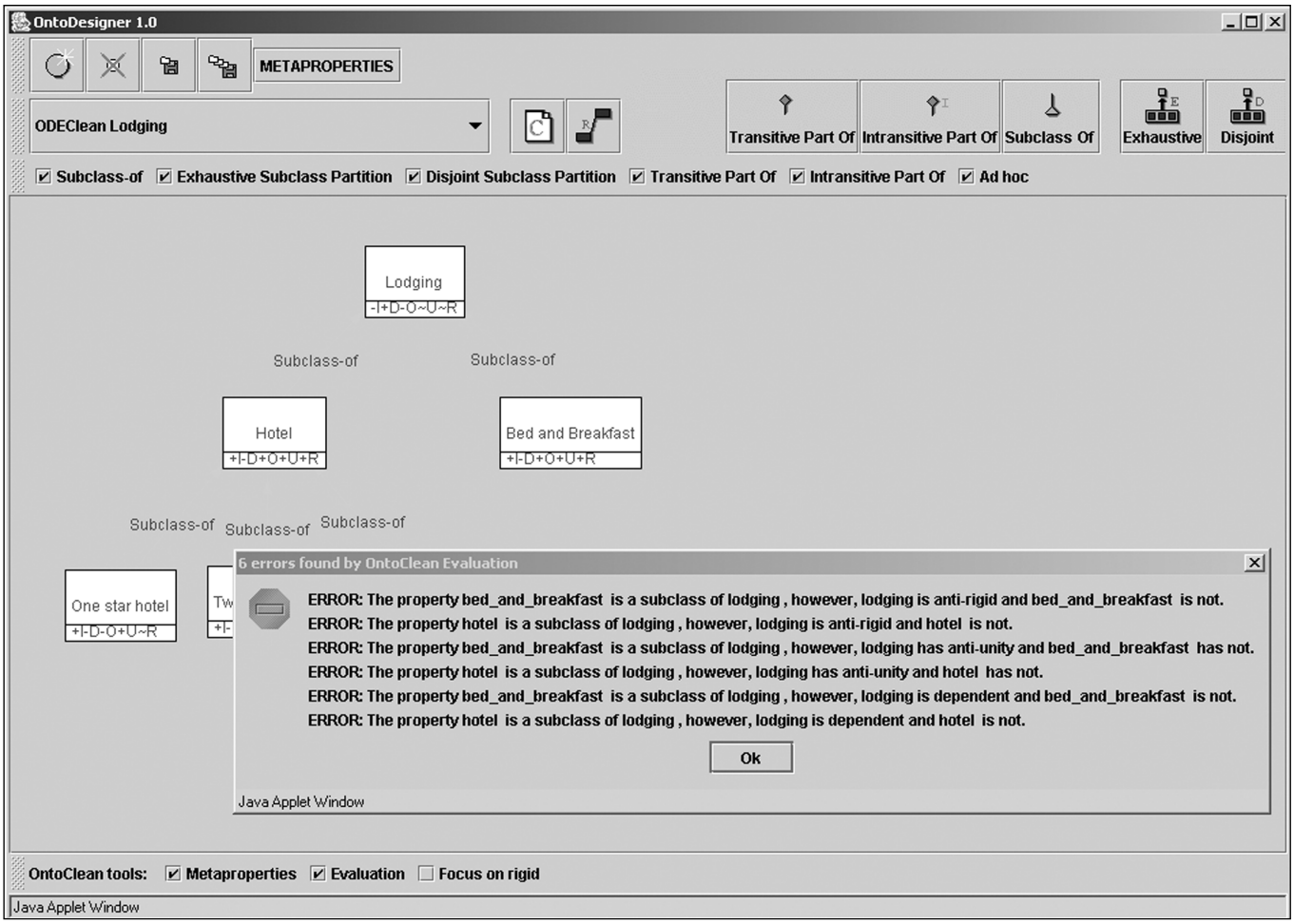


Figure 8. Ontology Evaluation with ODECLEAN.

implementation languages. Moreover, first-order logic axioms and rules can easily be created with WAB.

WEBODE has been built to support an ontology development methodology—METHONTOLOGY—but it does not prevent WEBODE from being used in the context of other methodologies for ontology development or without any specific methodological approach. It also provides a service that gives support to the ONTOCLEAN method for ontology evaluation.

WEBODE ontologies can be used in ontology-based applications in many different ways: (1) using its JAVA API directly, (2) using with RMI or web services from remote applications, (3) using XML files, (4) exporting and importing ontologies in different ontology languages from and into the workbench, (5) exporting ontologies to general languages such as PROLOG or JESS, and (6) exporting ontologies to JAVA.

The WEBODE workbench was built and tested in the context of the following projects, where we also developed several ontologies:

In the Spanish Comisión Interministerial de Cienca y Tecnolgía (CICYT) project on Methodology for Knowledge Management (TIC-980741), we built ontologies that modeled institutions and developed the core services of the workbench as well as the ODEKM application.

In the Spanish CICYT project CONTENTWEB (TIC2001-2745), we created WEBPICKER to perform ontology learning from electronic-commerce (e-commerce) standards for product and service classification (UNSPSC, ROSETTANET, and E-CL@SS). We also built ontologies in the domain of leisure activities.

In the UPM project Environment Ontology (UPM-AM-9819), we built ontologies in the domain of chemistry—elements and environmental ions—and integrated them with existing ontologies, such as the ONTOLINGUA ontology STANDARD UNITS.

In MRO (ontologies for cataloging business services), we merged heterogeneous electronic catalogs in the domain of office furniture.

Useful URLs

Ontology Tools

OilEd: oiled.man.ac.uk/

OntoEdit: www.ontoprise.de/products/ontoedit

ontolingua: ontolingua.stanford.edu

OntoSaurus: www.isi.edu/isd/ontosaurus.html

protege-2000: protege.stanford.edu/

WebODE: webode.dia.fi.upm.es/

WebOnto: kmi.open.ac.uk/projects/webonto/

WebODE Applications

ODESeW: webode.dia.fi.upm.es/sew/index.html

OntoRoadMap: webode.dia.fi.upm.es/ontoweb/wp1/OntoRoadMap/index.html

odekm: forgy.dia.fi.upm.es/km/login.html

mkbeem: www.mkbeem.com/

OntoWeb Special Interest Group on Enterprise-Standard Ontology Environments: delicias.dia.fi.upm.es/ontoweb/sig-tools/index.html

Ontology Markup Languages

rdf(s): www.w3.org/RDF/

oil: www.ontoknowledge.org/oil/

daml + oil: www.daml.org/language/

owl: www.w3.org/TR/owl-ref/

The WebODE workbench has been used to develop ontologies and ontology-based applications in the following projects:

In the European project Multilingual Knowledge-Based European Electronic Marketplace (MKBEEM) (IST-1999-10589), we built and reengineered business-to-business (B2B) and business-to-consumer (B2C) ontologies. These ontologies have been used in an e-commerce platform using the xcarin translator.

In the European thematic network ontoweb (IST-2000-29243), we built the OntoRoadMap application, which uses the WebODE API to access the ontologies and allows users to register, browse, and search ontologies, methodologies, tools, and languages for building ontologies as well as ontology-based applications.

In the European project Esperonto (IST-2001-34373), we built ODESeW, which is an ontology-based web portal that automatically creates an intranet and an extranet from WebODE ontologies. In this project, ODESeW is being used with ontologies that model a research and development project, including organizations and people participating in the project, documentation, work packages, and so on.

Finally, and also in the context of the European project Esperonto (IST-2001-34373), we are creating more middleware services and more ontology development and management services, such as ontology configuration management and evolution capabilities, ontology upgrading, semantic annotation services, ontology learning, and more integrated ontology reasoning services. We are laying the foundation for a more complete implementation of the workbench presented in the first section.

Acknowledgments

This work is supported by grant AP2002-3828 funded by Ministerio de Educación y Cultura and by the projects ContentWeb (TIC-2001-2745), mkbeem (IST-1999-10589), OntoWeb (IST-2000-29243), and Esperonto (IST-2001-34373). This workbench could not have been created without the help of R. de Diego, R. González, M. Lama, A. López, V. López, J. P. Pérez, E. Raya, and O. Vicente in the implementation and tests of WebODE services.

Notes

1. webode.dia.fi.upm.es/.

2. Ontology Building: A Survey of Editing Tools, www.xml.com/pub/a/2002/11/06/ontologies.html.

3. www.w3.org/TR/SOAP/.

4. www.w3.org/TR/wsdl.

5. www.clip.dia.fi.upm.es/Software/Ciao/index.-html.

References

Bechhofer, S.; Horrocks, I.; Goble, C.; and Stevens, R. 2001. OilEd: A Reasonable Ontology Editor for the Semantic Web. In *Joint German/Austrian Conference on Artificial Intelligence (KI'01)*, eds. F. Baader, G. Brewka, and T. Eiter, 396–408. Lecture Notes in Artificial Intelligence 2174. Vienna, Austria: Springer-Verlag.

Chaudhri, V. K.; Farquhar, A.; Fikes, R.; Karp, P. D.; and Rice, J. P. 1997. The Generic Frame Protocol 2.0. Technical Report, Knowledge Systems Laboratory, Stanford University.

Corcho, O.; Fernández-López, M.; Gómez-Pérez, A.; and Vicente, O. 2002. WebODE: An Integrated Workbench for Ontology Representation, Reasoning, and

Exchange. Paper presented at the Thirteenth International Conference on Knowledge Engineering and Knowledge Management (EKAW2002), 1–4 October, Sigüenza, Spain.

Domingue, J. 1998. TADZEBAO and WEBONTO: Discussing, Browsing, and Editing Ontologies on the Web. Paper presented at the Eleventh International Workshop on Knowledge Acquisition, Modeling, and Management (KAW'98), 18–23 April, Banff, Canada.

Duineveld, A.; Stoter, R.; Weiden, M. R.; Kenepa, B.; and Benjamins, V. R. 1999. Wonder Tools? A Comparative Study of Ontological Engineering Tools. Paper presented at the Twelfth Workshop on Knowledge Acquisition, Modeling, and Management (KAW99), 16–21 October, Banff, Canada.

Farquhar, A.; Fikes, R.; and Rice, J. 1997. The ONTOLINGUA Server: A Tool for Collaborative Ontology Construction. *International Journal of Human Computer Studies* 46(6): 707–727.

Fernández-López, M., and Gómez-Pérez, A. 2002. The Integration of ONTOCLEAN in WEBODE. Paper presented at the EKAW2002 Workshop on Evaluation of Ontology-Based Tools (EON2002), 30 September, Sigüenza, Spain.

Fernández-López, M.; Gómez-Pérez, A.; Pazos, J.; and Pazos, A. 1999. Building a Chemical Ontology Using METHONTOLOGY and the Ontology Design Environment. *IEEE Intelligent Systems and Their Applications* 4(1): 37–45.

Guarino, N., and Welty, C. Evaluating Ontological Decisions with ONTOCLEAN. *Communications of the ACM* 45(2): 61–65.

Noy, N. F.; Fergerson, R. W.; and Musen, M. A. 2000. The Knowledge Model of PROTEGE-2000: Combining Interoperability and Flexibility. Paper presented at the Twelfth International Conference in Knowledge Engineering and Knowledge Management (EKAW-00), 2–6 October, Juan-Les-Pins, France.

Studer, R.; Benjamins, V. R.; and Fensel, D. 1998. Knowledge Engineering: Principles and Methods. *IEEE Transactions on Data and Knowledge Engineering* 25(1–2): 161–197.

Sure, Y., and Studer, R. 2002. On-to-Knowledge Methodology. In *On-to-Knowledge: Semantic Web–Enabled Knowledge Management,* eds. J. Davies, D. Fensel, and F. van Harmelen, 33–46. New York: Wiley.

Sure, Y.; Erdmann, M.; Angele, J.; Staab, S.; Studer, R.; and Wenke, D. 2002. ONTOEDIT: Collaborative Ontology Engineering for the Semantic Web. In *Proceedings of the First International Semantic Web Conference (ISWC'02),* eds. I. Horrocks and J. A. Hendler, 221–235. Lecture Notes in Computer Science 2342. Berlin, Germany: Springer-Verlag.

Swartout, B.; Ramesh, P.; Knight, K.; and Russ, T. 1997. Toward Distributed Use of Large-Scale Ontologies. Paper presented at the Spring Symposium on Ontological Engineering, 24–26 March, Stanford, California.

Julio César Arpírez has a BA in computer science (2000) from the Universidad Politécnica de Madrid (UPM). He is currently a Ph.D. student in AI at UPM. He is currently working for a startup company, focusing on the 2.5G and 3G markets. His main interests include AI and enterprise and telecommunication systems. His e-mail address is jarpirez@delicias.dia.fi.upm.es.

Oscar Corcho has a BA in computer science (2000) and an MSc in software engineering (2001) from the Universidad Politécnica de Madrid (UPM). He is currently a Ph.D. student in AI at Universidad Politécnica de Madrid (UPM). He received the third Spanish award for computer science from the Spanish government in 2001. His research activities include ontology languages, ontology development tools, the ontology translation problem, and the semantic web. He is participating in the ONTOWEB thematic network (IST-2000-25056) and the IST Esperonto project (IST-2001-34373). His e-mail address is ocorcho@fi.upm.es.

Mariano Fernández-López has a BA in computer science (1996), an MSc in knowledge engineering (2000), an MSc in software engineering (2000), and a Ph.D. in computer science (2001), all from the Universidad Politécnica de Madrid (UPM). He is an associate professor in the Computer Science School at UPM and an associate professor at the Technical School of Fundación Universitaria San Pablo CEU. His research activities include ontology development and merging methodologies and ontology tools. His e-mail address is mfernandez@fi.upm.es.

Asunción Gómez-Pérez has been the director of the Ontology Group at the Universidad Politécnica de Madrid (UPM) since 1995. She has a BA in computer science (1990), an MSc in knowledge engineering (1991), and a PhD in computer science (1993) from the Universidad Politécnica de Madrid (UPM). She also has an MBA (1994) from Universidad Pontificia de Comillas. She is currently a research adviser at the AI lab at UPM. Her current research activities include interoperability between different kinds of ontology development tools, methodologies and tools for building and merging ontologies, ontological reengineering, ontology evaluation, and ontology evolution as well as uses of ontologies in applications related to the semantic web, e-commerce, and knowledge management. Her e-mail address is asun@fi.upm.es.

Call for Proposals

AAAI-04 Tutorial Forum

Nineteenth National Conference on Artificial Intelligence
July 25-29 San Jose, California
Sponsored by the American Association for Artificial Intelligence

The AAAI-04 Program Committee invites proposals for the Tutorial Forum of the Nineteenth National Conference on Artificial Intelligence (AAAI-04). The Tutorial Forum will be held July 25–29, 2004 in San Jose, California. Anyone interested in presenting a tutorial at AAAI-04 should submit a proposal to Matthew Stone, 2004 Tutorial Forum Chair, at the address below.

What Is the Tutorial Forum?

The Tutorial Forum provides an opportunity for junior and senior researchers to spend two days each year freely exploring exciting advances in disciplines outside their normal focus. We believe this type of forum is essential for the cross fertilization, cohesiveness, and vitality of the AI field. We all have a lot to learn from each other; the Tutorial Forum promotes the continuing education of each member of the AAAI.

Topics

AAAI is interested in proposals for advanced tutorials at the leading edge of AI. We especially encourage tutorials taught by a strong team of two established researchers, providing a balanced perspective on a topic of broad potential interest across the AI community. We are interested in tutorials that explore a specific research area in depth by summarizing its recent technical and methodological advances, educating participants about its important outstanding challenges and opportunities, and exploring its general ramifications for AI practice, including training as well as research. Our goal is to present a diverse program that includes core areas of AI, new techniques from allied disciplines that can inform research within AI, and conversely emerging applications of AI techniques to new areas. Previous years' tutorial programs provide an indication of the scope and variety of possible topics. The list is not exclusive; indeed, we are expressly interested in topics that we would not have imagined to mention. Finally, note that we very much welcome proposals for educational approaches that go beyond the traditional format of four-hour tutorials, exploiting the flexibility that the one-fee program offers.

AAAI-02's forum included tutorials on language modeling; AI in space; greedy on-line planning; qualitative spatial and temporal reasoning; practical approaches to handling uncertainty in planning and scheduling; collaborative multi-agent systems; practical machine learning for software engineering; information integration on the web; AI techniques for personalized recommendation; algorithms for combinatorial auctions and exchanges; phase transitions and structure in combinatorial problems; and rational action in autonomous agents. This list serves merely as an example. We are looking for continued innovation in the forum's program that incorporates novel and under-represented topic areas.

Submission Requirements

We need two kinds of information in the proposals: information that will be used for selecting proposals and information that will appear in the tutorial description brochure. The proposal should provide sufficient information to evaluate the quality of the technical content being taught, the quality of the educational material being used, and the speakers' skill at presenting this material. Each proposal should include at least the following:

- *Goal of the tutorial:* Who is the target audience? What will the audience walk away with? What makes the topic innovative?
- *Content:* Detailed outline and list of additional materials, augmented with samples, such as past tutorial slides and survey articles, whenever possible. Be as complete as possible.
- *Tutorial description:* A short paragraph summarizing the tutorial outline.
- *Prerequisite knowledge:* What knowledge is assumed.

Please also submit the following information about the team of presenters: name, mailing address, phone number, e-mail address; background in the tutorial area, including a list of publications and/or presentations; any available examples of work in the area (ideally, a published tutorial-level article or presentation materials on the subject); evidence of teaching experience (courses taught or references); and evidence of scholarship in AI or computer science.

Submission Deadline

Proposals must be received by October 15, 2003. Decisions about the tutorial program will be made by December 1, 2003. Speakers should be prepared to submit completed course materials by May 25, 2004.

Please e-mail proposal material to the tutorial chair at the following address. Hard copy submissions will also be accepted:

- Matthew Stone
  Department of Computer Science — Rutgers University
  110 Frelinghuysen Road
  Piscataway NJ 08854-8019
  Telephone: 732 445 3546 / Fax: 732 445 1494
  E-mail: mdstone@cs.rutgers.edu