Reference Ontology and (ONTO)^{2}Agent: The Ontology Yellow Pages

Julio César Arpírez[1], Asunción Gómez-Pérez[1],
Adolfo Lozano-Tello[2] and Helena Sofia Andrade N. P. Pinto[3]

[1]Facultad de Informática, Universidad Politécnica de Madrid, Madrid, Spain
[2]Departamento de Informática, Universidad de Extramadura, Extremadura, Spain
[3]Departamento de Engenharia Informática, Instituto Superior Técnico, Lisbon, Portugal

Abstract. Knowledge reuse by means of ontologies faces three important problems at present: (1) there are no standardized identifying features that characterize ontologies from the user point of view; (2) there are no web sites using the same logical organization, presenting relevant information about ontologies; and (3) the search for appropriate ontologies is hard, time-consuming and usually fruitless. To solve the above problems, we present: (1) a living set of features that allow us to characterize ontologies from the user point of view and have the same logical organization; (2) a living domain ontology about ontologies (called *Reference Ontology*) that gathers, describes and has links to existing ontologies; and (3) (ONTO)^{2}Agent, the ontology-based WWW broker about ontologies that uses Reference Ontology as a source of its knowledge and retrieves descriptions of ontologies that satisfy a given set of constraints.

1. Introduction and Motivation

During recent years, considerable progress has been made in developing the conceptual bases for building technology that allows knowledge component reuse and sharing. One of the main motivations underlying both ontologies and problem-solving methods (PSM) is to enable sharing and reuse of knowledge and reasoning behavior across domains and tasks. PSMs and ontologies can be seen as complementary reusable components to construct knowledge systems (Gómez-Pérez

and Benjamins, 1998). Ontologies are concerned with static domain knowledge and PSMs with dynamic reasoning knowledge. The integration of ontologies and PSMs is a possible solution to the 'interaction problem' (Bylander and Chandrasekaran, 1988), which states that representing knowledge for the purpose of solving some problem is strongly affected by the nature of the problem and the inference strategy to be applied to the problem.

Ontologies are defined as a formal, explicit specification of a shared conceptualization (G. Gruber, 1993; Borst, 1997); that is, 'Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used, and the constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine-readable. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group' (Studer et al, 1998). PSMs describe the reasoning process of a knowledge-based system in an implementation- and domain-independent manner (Benjamins and Fensel, 1998). There are also the notions of task ontologies (Mizoguchi et al, 1995) and PSM ontologies (Chandrasekaran et al, 1998).

Nowadays, it is easy to get information from organizations that have ontologies and PSMs on the web. There are even accessible points that gather information about ontologies and have links to other web pages containing more explicit information about such ontologies (see The Ontology Page, also known as TOP; http://www.medg.lcs.mit.edu/doyle/top) and there are also ontology servers, like The Ontology Server (http://www-ksl.standford.edu:5915) (Farquhar et al, 1995, 1996), Cycorp's Upper CYC Ontology Server (http://www.cyc.com) (Lenat, 1990) or Ontosaurus (http://indra.isi.edu:8000/Loom) (Swartout et al, 1997), that collect a huge number of very well-known ontologies. In the PSM area, there are also many PSM repositories at different locations but they are not accessible for outsiders and they are not compatible (Benjamins et al, 1998).

At present, the knowledge component reuse and sharing community has identified the need to provide intelligent agents or intelligent brokering services on the WWW that ease the search for such knowledge components. In the ontology field, the need for this kind of services was identified in Fikes and Farquhar (1997) and Foundation for Intelligent Physical Agents (1998), but there are no web sites that gather information about ontologies that have already been built using the same logical organization, and there are no intelligent brokers specializing in the ontology field that could help in this search. This paper presents $(ONTO)^2$Agent, an ontology-based WWW broker that helps to select ontologies. There is a similar project in the PSM field, called IBROW3 (Benjamins et al, 1998), whose goal is to develop an intelligent brokering service for PSM reuse on the WWW.

Apart from the problems arising from component search, the choice of a knowledge component that does not match the system needs properly, whose usage is expensive (people, hardware and software resources, time), or that has not been satisfactorily evaluated technically (verified and validated) may force future users to stop reusing the built knowledge component and oblige them to formalize the same knowledge again.

These problems are probably what have led to there being relatively few applications known to date in areas like knowledge management, ontology-based brokers, natural language generation, enterprise modeling, knowledge-based systems, and interoperability between systems that reuse ontologies.

This paper presents a solution to the problem of locating and searching the appropriate ontology on the web. End users usually face a complex multi-

criteria choice problem when they search for candidate ontologies. Apart from the dispersion of ontologies over several servers:

(a) Ontology content formalization differs depending on the server at which it is stored.
(b) Ontologies on the same server are usually described with different detail levels.
(c) There is no common format for presenting relevant information about ontologies so that users can decide which ontology best suits their purpose.

To speed up the use of ontologies in applications, our solution is to prepare a kind of *yellow pages of ontologies* that provide classified and updated information about ontologies following the same logical organization. These living[1] yellow pages help future users to locate candidate ontologies for a given application. (ONTO)^{2}Agent, the *intelligent WWW broker* specialized in the ontology field that uses an ontology as its knowledge source, spreads information about existing ontologies, helps to search appropriate ontologies, reduces the search time for the desired ontology, and supplies pointers to the set of ontologies that totally/partially meet user requirements.

This paper is organized as follows. In Section 2, we present an overview of our solution. Section 3 presents a set of features to characterize, compare, evaluate and assess ontologies from the user point of view. Section 4 shows how we have built the Reference Ontology (an ontology in the domain of ontologies) to be used by the broker specialized in the ontology field. Finally, Section 5 presents the OntoAgent architecture, that is, the technology we use to build ontology-based WWW brokers and how it has been instantiated in (ONTO)^{2}Agent, the broker that answers questions in the domain of ontologies using the Reference Ontology as its knowledge source. (ONTO)^{2}Agent is capable of answering questions about the features of ontologies that have been entered into the Reference Ontology. A specimen query would be: give me all the ontologies in the domain D that are implemented in languages L1 and L2.

2. An Overview

Three main tasks were identified in our approach:

1. Characterization of domain knowledge, which, in our case, is the ontology field, by identifying a set of features that allows end users to compare different ontologies.
2. Ontological Engineering, to build the ontology, which, in our case, is the Reference Ontology, a domain ontology in the domain of ontologies using the METHONTOLOGY framework (Fernández et al, 1997, 1999; Gómez-Pérez, 1996) and the Ontology Design Environment (Blázquez et al, 1998).
3. Intelligent brokering services to access the ontology content, which, in our case, is the OntoAgent architecture, that is, the technology we have used to build ontology-based WWW brokers.

Figure 1 shows how these tasks are related. The development of the ontology was divided into two phases. The first phase is centralized and concerns the

[1] Living in the sense that ontologies evolve and grow, as presented in Tennison and Shadbolt (1998).

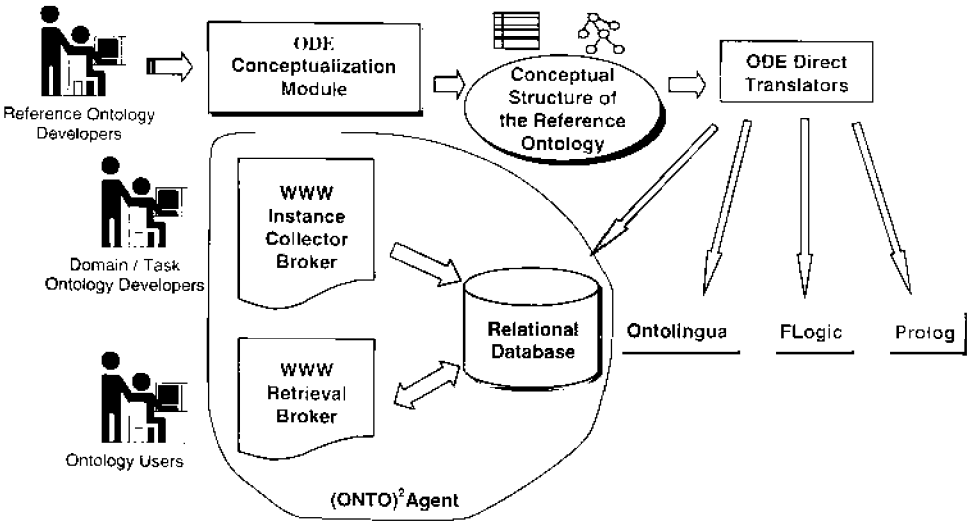

Fig. 1. General overview of (ONTO)^{2}Agent development process.

development of the conceptual structure of the ontology, and the identification of its main concepts, taxonomies, relations, functions and axioms. The second phase is distributed among ontology developers and involves filling in WWW forms to enter knowledge about instances into the conceptual structure. The ontology (conceptual structure+instances) is centralized and stored in a relational database. Using a WWW interface, the user can consult the ontology using the domain ontology vocabulary. (ONTO)^{2}Agent searches for instances that satisfy the query and provides the answer.

3. Features for Comparing Ontologies

3.1. Previous Work

Although software engineering and knowledge engineering provide detailed features for evaluating the fitness of components and calculating their reusability in software applications (Slagle and Wick, 1988; Basili et al, 1994; Kan, 1995; Khairuddin and Key, 1995; Pressman, 1997), the literature reviewed in the field of ontologies shows that there are few papers on identifying features for describing, comparing and assessing ontologies. There are some articles about ontology features (e.g., Fridman and Hafner, 1997; Hovy, 1997; Arpírez, 1998; Uschold, 1998), and there are articles from which they can be extracted, although they do not specifically address ontology features (Uschold and Grüninger, 1996). These papers are presented in chronological order below.

One of the first articles from which we can extract general features to characterize ontologies is Uschold and Grüninger (1996). The features identified are: formality, purpose and domain. Formality expresses the level of formality of one ontology. Its possible values are: informal, semi-informal, semi-formal and highly formal. Purpose characterizes the different uses of one ontology. The values

proposed are: communication, interoperability and systems engineering. Domain describes the piece of reality that the builder of the ontology wants to represent. The problem with this proposal is that it specifies only a few features and, hence, ontology characterization is limited.

Another article presenting features for comparing ontologies is Fridman and Hafner (1997). In this proposal, features are classified as: general, related to the design process, related to the taxonomy, related to the internal concept structure and the relations between concepts, related to axioms and the inference mechanism, related to the applications built using the ontology, and important contributions. General features try to give an overview of the ontology; for instance, whether the ontology is general or domain-specific, its domain (if it is a domain-specific ontology), the size of the ontology, the formalism used to represent it, purpose, etc. Design process features characterize the building process and whether the ontology was evaluated. Taxonomy features characterize the structure of the ontology and the main concepts represented in the ontology. Internal structure and relations features characterize the concepts represented and the relations used to represent those concepts. Axioms and inference features characterize the kind of logic used to represent and infer knowledge. Application describes the applications built using the ontology. Contributions refer to the strengths and weaknesses of the ontology. This taxonomy of 25 features was used to compare CYC (Lenat, 1990), Wordnet (Miller, 1990), GUM (Bateman et al, 1994), Sowa's Ontology (Sowa, 1997), Dahlgren's Ontology (Dahlgren, 1988), UMLS (Humphreys and Lindberg, 1993), TOVE (Gruninger and Fox, 1995), GENSIM (Karp, 1993), Plinius (Van der Vet et al, 1994) and KIF (Genesereth and Fikes, 1992). The problems with this proposal are the limited characterization of ontologies, and the lack of criteria for classifying ontologies according to some of the features.

Proposed at the same time as Fridman and Hafner (1997), Hovy (1997) presents a taxonomy of features for comparing ontologies for natural language processing. This proposal classifies its 36 features into form, content and usage. Form features characterize the conceptual tools with which the ontology builder defines terms and axioms. They are related to mathematical logic and computational complexity. Some of the most important features according to this criterion are: parsimoniousness, type flipping and coercion, inheritance of properties, support of inferences, worlds or contexts, TMS and non-first-order reasoning. Content features refer to terminology, axiom inferences and instances used by the ontology builder. They are related to philosophy, epistemology and the domain. Some of the most important features covered by this criterion are: size, organization, average specificity (general or domain-specific), linkage to other resources, characterization of axioms and inferences, instance fidelity and instance coverage. Finally, usage features describe the way in which the ontology builder builds, uses and maintains the ontology. They are related to software engineering and ergonomics. Some of the most important features covered by this criterion are: storage size, hardware and software platforms required, viewing and editing tools and documentation. The major weakness of this proposal is that it is incomplete. Although it covers a considerable number of features, some of the features are not defined, some are not clearly defined, some definitions contradict the examples and there are several features that are only relevant to natural language issues.

Uschold (1998) proposes 10 features for classifying ontology applications: purpose, representation language and paradigms, meaning and formality, subject matter (domain), scale, development (is the ontology implemented, published,

planned?), conceptual architecture (main components of the application), mechanisms and techniques (to use the ontology), implementation platform (hardware and software required), and miscellaneous (hidden assumptions). The main drawback of this proposal is that ontology characterization is limited.

Although our initial proposal of 70 features (Arpírez et al, 1988), was published at the same time as Uschold's (1998), the purpose of the features we propose is quite different from the purposes of previous (Fridman and Hafner, 1997; Hovy, 1997; Uschold and Grüninger, 1996) and simultaneous (Uschold, 1998) proposals: we want to characterize ontologies from the user point of view.

3.2. Characterization of Ontologies

The goal of this section is to provide a wide initial set of features that allow us to characterize the ontologies from the user point of view by identifying the main attributes and their values. The kinds of questions we are trying to answer are, for example: In which languages is the ontology available? What are the mechanisms for interacting with the ontology? Is the knowledge represented in a frame-based formalism? What is the cost of the hardware and software infrastructure needed to use the ontology? What is the cost of the ontology? Is the ontology well documented? Was it evaluated (Gómez-Pérez, 1996) from a technical point of view?

To be able to answer the above questions, we undertook a detailed study of the ontologies available at ontology servers on the web (Ontology Server, Cyc Server, Ontosaurus) and other ontologies from the literature (PhysSys: Borst et al, 1996; Borst, 1997; EngMath: Gruber and Olsen, 1994). Our aim is twofold: first, to identify the most representative features of these ontologies (developers, ontology server, type, purpose, etc.); second, to define a shared domain ontology about ontologies (the Reference Ontology, in Section 4) and relate each individual ontology to that shared ontology. This Reference Ontology can help future users to select the most adequate and suitable ontology for the application they have in mind.

To improve and speed up the process of searching for the features of the ontology, they are grouped into the following categories: identifying, descriptive and functional features, as shown in Fig. 2. A preliminary set of features is proposed for each category. Since not all the features are equally important, the essential features, that is, features which are indispensable in order to distinguish each ontology, are given in bold type. It is compulsory to fill in these features. We also stress that: (1) some features cannot be used to characterize certain ontologies; (2) the ontology builder may not know the values of some features; (3) this is an evolving list of features to be improved and completed with new features as and when required; and (4) several features can be interrelated, but the characteristics are defined and categorized to find suitable ontologies easily and intuitively.

3.2.1. Identifying Features

These provide information about the ontology itself, its developers and distributors. We consider it important to specify the following:

- *About the ontology*: its name, server site, mirror sites, web pages, FAQs available, mailing lists, natural language description and date when the ontology was built.
- *About the main developers and distributors*: their names, web pages, emails, contact names, telephone and fax numbers and postal addresses.

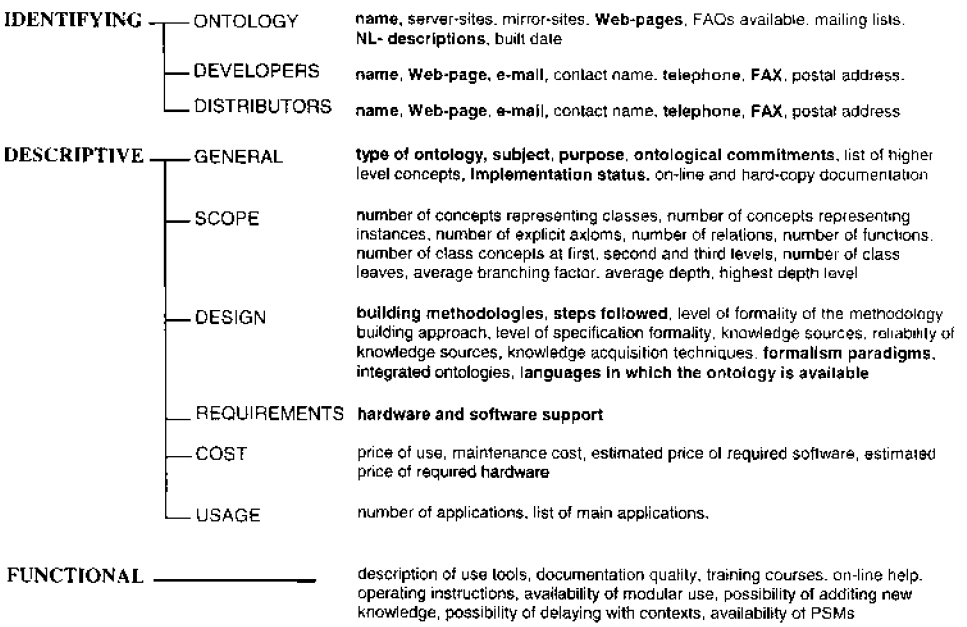

Fig. 2. Features for comparing ontologies.

3.2.2. Descriptive Features

These provide information about the content and form of the ontology. They have been divided into six categories: general, scope, design, requirements, cost and usage.

General features describe basic content issues. Users will frequently consult this kind of information, since these features are crucial for looking up other features. We considered the following properties: type of ontology (as proposed in van Heist et al, 1997, the possible values are application, domain, generic, representation), subject of the ontology, purpose (we identified modeling, NL understanding, knowledge sharing and reuse, simulation, teaching, theoretical research, others, as possible values), ontological commitments (T. Gruber, 1993), list of higher-level concepts, implementation status, and on-line and hard-copy documentation.

Scope features describe measurable attributes proper to the ontology. They give an idea of the content and depth of the ontology. Properties to be taken into account are: number of concepts representing classes, number of concepts representing instances, number of explicit axioms, number of relations and functions, number of class concepts at first, second and third levels, number of class leaves, average branching factor, average depth, highest depth level.

Design features describe the method used to build the ontology, the activities carried out during the whole process and how knowledge is organized and distributed in the ontology.[2]

[2] The ontology can be divided into several ontologies.

1. It is important to mention the methodology used, the steps (Fernández et al, 1997; Blázquez et al, 1998; Gómez-Pérez, 1998) taken to build the ontology (mainly planning, specification, knowledge acquisition, conceptualization, integration, implementation, evaluation, documentation and maintenance) according to the selected methodology, its level of formality (as proposed in; informal, semi-informal, semi-formal, highly formal), and the construction approach (as proposed in Uschold and Grüninger, 1996; top-down, middle-out and bottom-up).
2. Depending on the methodology, the requirement specification may be formal, informal or semi-formal (Gómez-Pérez, 1998).
3. With regard to knowledge acquisition, it is important to state the types of knowledge sources (we identified domain books, experts, ontologies, standards, others, as possible values), how reliable such knowledge sources are (we identified three levels of reliability: high, average and low) and the techniques used in the process (formal and/or informal text analysis, interviews, protocol analysis, repertory grids and others).
4. With respect to formalism paradigms, a frame-based formalism, a first-order logic approach, a semantic network, like conceptual graphs, or even a hybrid knowledge representation paradigm can be selected (the list of possible values is: classical logic, non-classical logic, non-monotonic logic, frame-based, semantic networks, hybrid systems and others). It is important to state here that the chosen formalism constrains the knowledge representation ontology in which a given ontology is implemented. For example, if we select a frame-based formalism paradigm, then the major candidate is the frame ontology at the Ontology Server. The formalism paradigm also plays a major role in ontology integration. For example, if you want to integrate an ontology built using a first-order language into a frame-based paradigm a lot of knowledge will be lost due to the weaker expressive power of the latter.
5. As far as integration is concerned, a list of the integrated ontologies should be given.
6. Finally, we need to know from the implementation point of view, the source languages in which the ontology is supplied (as for instance: Ontolingua, G. Gruber, 1993; F-Logic, Kifer et al, 1995; LOOM, MacGregor, 1991; Lisp, Prolog, Relational Database) and the list of formal KR languages supported by available translators.

Requirement features identify the minimum hardware (swap and hard disk space, RAM, processor, operating system) and software support requirements (knowledge representation languages and implementation language underneath the KR language) for using the ontology. All these features will greatly influence costs.

Cost features help to assess the estimated cost of using the ontology in a given organization. Since hardware and software costs vary widely and depend on the existing computer infrastructure, the total cost should be calculated by adding the cost of use and maintenance to the features identified above (estimated prices of the hardware and software required).

The usage features refer to the applications that use this ontology as a source of their knowledge. The number of known applications and their names are the features to be filled in by the informant.

3.2.3. *Functional Features*

These properties give clues as to how the ontology can be used in applications. By observing these features, users will be able to estimate the technical and operational effort that will be required if they decide to reuse the ontology. We identified the following features: description of use tools (taxonomic browsers, editors, evaluators, translators, remote access modules, etc.), quality of documentation (we identified five levels of quality: excellent, good, average, marginal, poor), training courses available, on-line help available, how to use the ontology (including the steps followed to access, manipulate, display and update knowledge from remote and on-site applications), availability of modular use, possibility of addition of new knowledge, possibility of dealing with contexts, availability of integrating PSMs, etc.

3.3. Discussion

Some of the above features can have different influence in the user requirements, and this influence depends on the particularities of the project or the company that will reuse the ontology. That is, in this paper, neither the particularities of the human and material resources of the company, nor the size and complexity of the project will be taken into account. However, all project managers should weigh up each category and features to decide how important they are, relating them to the peculiarities and demands of their particular project.

4. Design of an Ontology about Ontologies: The Reference Ontology

Having presented a living set of features that describe each ontology and differentiate one ontology from another, the goal of this section is to show how we have built the Reference Ontology using the features identified in Section 3. As stated above, the Reference Ontology is a domain ontology about ontologies that plays the role of a kind of yellow pages of ontologies. Its aims are to gather, describe and have links to existing ontologies, using a common logical organization.

The development of the Reference Ontology was divided into two phases. The first phase is concerned with the development of its conceptual structure, and the identification of its main concepts, taxonomies, relations, functions and axioms. It was carried out using the METHONTOLOGY framework (Fernández et al, 1997; Gómez-Pérez, 1998; Fernández et al, 1999) and the Ontology Design Environment (Blázquez et al, 1998). As one of the research topics of the $(KA)^2$Ontology (Benjamins et al, 1999) is ontologies, we decided to incorporate the Reference Ontology into the Product ontology of the $(KA)^2$ initiative that is currently being developed by the KA community. The second phase corresponds to the addition of knowledge about specific ontologies that act as instances in this Reference Ontology. Ontology developers will enter such knowledge using a WWW form also based on the features previously presented in Section 3. Thus, the effort made to collect information about specific ontologies is distributed among ontology developers while the development of the conceptual structure of the Reference Ontology is centralized.

4.1. Reference Ontology Conceptual Structure

4.1.1. Sources of Knowledge

As starting points for developing our Reference Ontology, we took three sources of knowledge. The first source was the set of features presented earlier in Section 3, which played the role of a specification document. The second source was the restructured $(KA)^2$ conceptual model (Blázquez et al, 1998). The third source was the set of properties identified for the Research Topic ontology, which were established during the KEML workshop held at Karlsruhe, on January 23, 1998. The properties identified were: Name, Description, Approaches, Research-groups, Researchers, Related-topics, Sub-topics, Events, Journals, Projects, Application-areas, Products, Bibliographies, Mailing-lists, WebPages, International-funding-agencies and National-funding-agencies. All these properties describe the field of ontologies and differentiate it from other fields of research. However, the properties we presented in Section 3 characterize each ontology and differentiate one ontology from another. Some of the features presented in Section 3 lead to some minor changes and extensions to the $(KA)^2$Ontology.

4.1.2. Conceptual Structure

The goal of conceptualization in the METHONTOLOGY framework is to organize and structure the acquired knowledge in a complete and consistent knowledge model, using external representations (glossary of terms, concept classification trees, *ad hoc* binary relation diagrams, concept dictionary, table of *ad hoc* binary relations, instance attribute table, class attribute table, logical axiom table, constant table, formula table, attribute classification trees and an instance table) that are independent of implementation languages and environments. As a result of this activity, the domain vocabulary is identified and defined. The following describes the process followed to build the Reference Ontology conceptual model and how it was integrated into the $(KA)^2$Ontology.

First, we built a *glossary of terms* that included all the terms (concepts, instances, attributes, verbs, etc.) of the domain and their description in natural language, as shown in Table 1.

When the glossary of terms contained a sizeable number of terms, we structured the domain knowledge in *concept classification trees* following a hierarchical organizational principle. This principle aims to divide the domain knowledge into as many independent modules as possible to which inheritance can be applied. Each taxonomy produced an ontology. From there, we integrated the Reference Ontology into the $(KA)^2$Ontology. Not only did we build concept classification trees for the Reference Ontology, but also the concepts identified in the glossary of terms of the Reference Ontology were integrated into the concept classification trees of the $(KA)^2$Ontology. These concepts appear in bold type in Fig. 3. For instance, the classes Servers and Languages are subclasses of Computer-Support in the Product ontology. The subclass of the class Servers is the class Ontology-Servers, whose instances (in italics) are the Ontology-Server, the Ontosaurus and the CycServer. The subclass of the class Languages is the class Ontology-Languages, whose instances are Ontolingua, CycL and LOOM.

The next step was to build *ad hoc binary relations diagrams* between concept classification trees. The goal of this diagram is to establish relationships between concepts of the same or different ontologies. Note that this diagram

Table 1. Glossary of terms in the Reference Ontology.

Name	Description
Ontology	An explicit specification of a conceptualization
Ontology-Languages	Languages used to represent ontologies
Ontology-Servers	Servers that support ontologies
Developed-by	The organization that developed the ontology
Purpose-of-Ontology	Original purpose for which the ontology was built and its intended uses
Server-Translates-to-Languages	The languages the server translates to
Is-Translated-from-Server	The servers that have translators to the language
Average-Branching-Factor	Average number of outgoing branches of a node
Ontology-Formalized-in-Language	Language/formalism used to represent the knowledge contained in the ontology
Ontolingua	A language specially defined to represent ontologies
List-of-Available-Languages	All formal knowledge representation languages in which the ontology has been formalized and which can be supported by available translators
...	...

will set out the guidelines for integrating ontologies, because if a concept C1 is linked by a relation R to a concept C2, this means that the ontology containing C1 includes the ontology containing C2, provided that C1 and C2 are in different concept classification trees. We have included new relations between classes in the *ad hoc binary relations diagrams* of the $(KA)^2$Ontology; for instance, the relation Ontology-Located-at-Server that relates an ontology to a server or the relation Ontology-Formalized-in-Language that relates an Ontology to an Ontology-Language.

For *each* concept classification tree generated, we built the following IRs:

1. A *concept dictionary* containing all the domain concepts, instances of such concepts, class and instance attributes of the concepts and, optionally, concept synonyms and acronyms. Table 2 sets out a small part of the concept dictionary of the Reference Ontology.
2. A table of binary relations for each *ad hoc* relation whose source concept is in the concept classification tree. For each relation, we specified: its name, the name of the source and target concept, its inverse relation, etc. Tables 3 and 4 show the Server-Translates-to-Languages relationship and its inverse Is-Translated-from-Server.
3. An *instance attribute table* for each instance attribute that appears in the concept dictionary. Instance attributes are attributes that are defined in the concept but that take values in its instances. For example, the Average-Branching-Factor of an ontology is proper to each instance. For each instance attribute, we included: its name; the value type; the measurement unit for numerical values; accuracy for numerical values; range of values; default values; minimum and maximum cardinality; instance attributes, class attributes and constants that are used to infer the value of the attribute that is being defined; attributes that can be inferred using this attribute; formula or rule for inferring the attribute that is being defined; and references used to fill in the attribute. Table 5 shows the attribute Average-Branching-Factor.

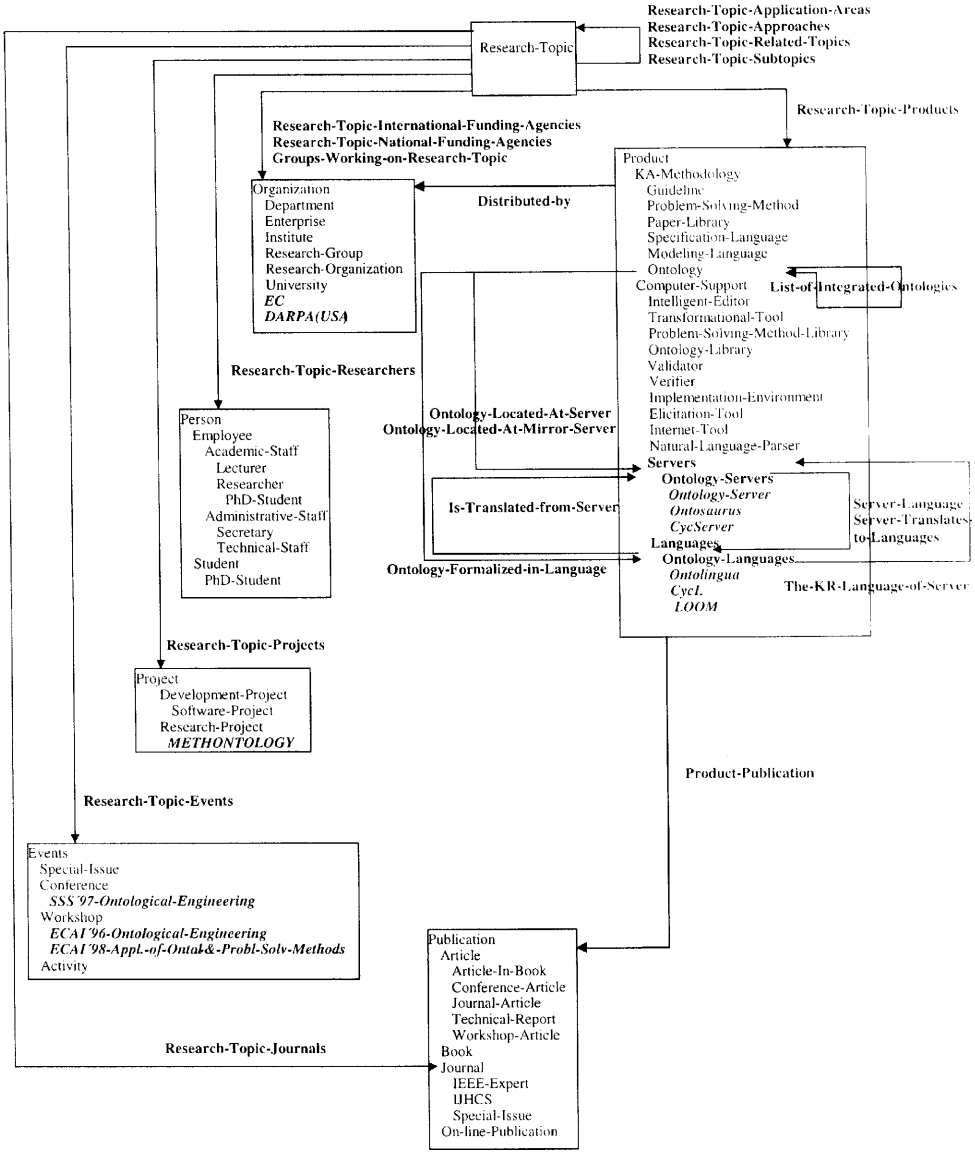

Fig. 3. Some of the relations and concepts added to the $(KA)^2$ ontology.

4. A *logical axioms table.* This is used to define the concepts by means of logical expressions that are always true. Each axiom defined included: its name, its natural language description, the concept to which the axiom refers, the attributes used in the axiom, the logical expression that formally describes the axiom using FOPC and references, as shown in Table 6.
5. An *instance table* for each instance that appears in the concept dictionary. Each table includes: the name of the instance, the attributes with known values in the instance and the values of such attributes. It is important to mention that

Table 2. Concept dictionary in the domain of the Reference Ontology.

Concept name	Instances	Class attributes	Instance attributes	Relations
Ontology	—	—	Type-of-Ontology	Ontology-Formalized-in-Language
			Subject-of-Ontology	
			Purpose-of-Ontology	Ontology-Located-at-Server
			Average-Branching-Factor	Ontology-Located-at-Mirror-Server
			...	List-of-Integrated-Ontologies
				List-of-Available-Languages
				...
Ontology-Languages	CycL	—	The-KR-Language-of-Server	—
	Ontolingua			
	LOOM		Is-Translated-from-Server	
	F-Logic		...	
	Relational-Data-Base			
	...			
...	...	...	...	...

Table 3. Binary relation: Server-Translates-to-Languages in the domain of the Reference Ontology.

Relation name	Server-Translates-to-Languages
Source concept	Servers
Source cardinality	(0, *n*)
Target concept	Languages
Mathematical properties	—
Inverse relation	Is-Translated-from-Server
References	

the Reference Ontology includes only instances related to ontology servers and languages used to implement ontologies. It does not gather knowledge about specific ontologies because this knowledge will be collected from the community, in a distributive way, using the WWW instance collector broker.

The conceptual structure of the Reference Ontology was built using the ODE

Table 4. Binary relation: Is-Translated-from-Server in the domain of the Reference Ontology.

Relation name	Is-Translated-from-Server
Source concept	Languages
Source cardinality	(0, *n*)
Target concept	Servers
Mathematical properties	—
Inverse relation	Server-Translates-to-Languages
References	

Table 5. Instance attribute: Average-Branching-Factor in the domain of the Reference Ontology.

Instance attribute name	Average-Branching-Factor
Value type	Real
Unit of measure	—
Precision	—
Range of values	[0, $+\infty$]
Default value	—
Cardinality	(1, 1)
Inferred from instance attribute	—
Inferred from class attribute	—
Inferred from constants	—
Formula	—
To infer	—
References	—

conceptualization module. ODE assures the consistency and completeness of the knowledge represented in each IR and between IRs. It is important to mention that the process of building the conceptual model using these IRs is not sequential in the sense of a waterfall life-cycle model, but some order must be followed so as to assure the consistency and completeness of the knowledge already represented.

Once the ontology had been conceptualized, we generated three implementations of this ontology using ODE translators: an Ontolingua implementation, an FLogic implementation and a relational database implementation. The broker used to select ontologies will use the latter implementation to search for the most appropriate ontology.

4.1.3. Design Criteria

This section presents the design criteria used to incorporate the Reference Ontology into the $(KA)^2$Ontology:

- Modularity: we sought to build a module-based ontology to allow more flexibility and varied uses.
- Specialize: we identified general concepts that were specialized into more specific concepts until domain instances were reached. Our goal was to classify concepts by similar features and to guarantee inheritance of such features.
- Diversify each hierarchy to increase the power provided by multiple inheritance

Table 6. Named axiom: translation of ontologies written in Ontolingua in the domain of the Reference Ontology.

Axiom name	Translation of ontologies written in Ontolingua
Description	Languages to which the translators available at the ontology server can translate a given ontology
Concept	Ontologies
Referred attributes	Ontology-Formalized-in-Language
Variables	X
Expression	For all(X) (Ontology(X) and Ontology-Formalized-In-Language(X, Ontolingua) ⇒ List-of-Available-Languages(X, Clips) and List-of-Available-Languages(X, ClipsSententialFormat) and List-of-Available-Languages(X, CML-ATP) and List-of-Available-Languages(X, CML-Rule-Engine) and List-of-Available-Languages(X, EPIKIT) and List-of-Available-Languages(X, IDL) and List-of-Available-Languages(X, KIF3.0) and List-of-Available-Languages(X, KSL-Rule-Engine) and List-of-Available-Languages(X, LOOM) and List-of-Available-Languages(X, OKBC-friendly) and List-of-Available-Languages(X, Prolog-Syntax))
Relations	—
References	—

mechanisms. If enough knowledge is represented in the ontology and as many different classification criteria as possible are used, it is easier to enter new concepts (since they can be easily specified from the pre-existing concepts and classification criteria).

- Minimize the semantic distance between sibling concepts: similar concepts are grouped and represented as subclasses of one class and should be defined using the same primitives, whereas concepts which are less similar are represented further apart in the hierarchy.
- Maximize relationships between taxonomies: *ad hoc* relations and slots were filled in as concepts in the ontology.
- We have not taken into account ontology server, ontology and language releases to build our ontology. For instance, in our ontology, the Ontology Server is an instance of servers and we do not keep records of its latest and future releases.
- Standardize names: whenever possible we specified that a relation should be named by concatenating the name of the ontology (or the concept representing the first element of the relation), the name of the relation and the name of the target concept; for instance, the relation Ontology-Formalized-in-Language between the class of ontologies and one Language.

All changes, the entry of new relations and properties and the entry of new concepts were guided by the features presented in Section 3. Essentially, the $(KA)^2$Ontology was extended using new concepts and some knowledge previously represented in the $(KA)^2$Ontology was specialized in order to represent the information that we found was of use and of interest for comparing different ontologies with a view to reuse or use as a basis for further applications.

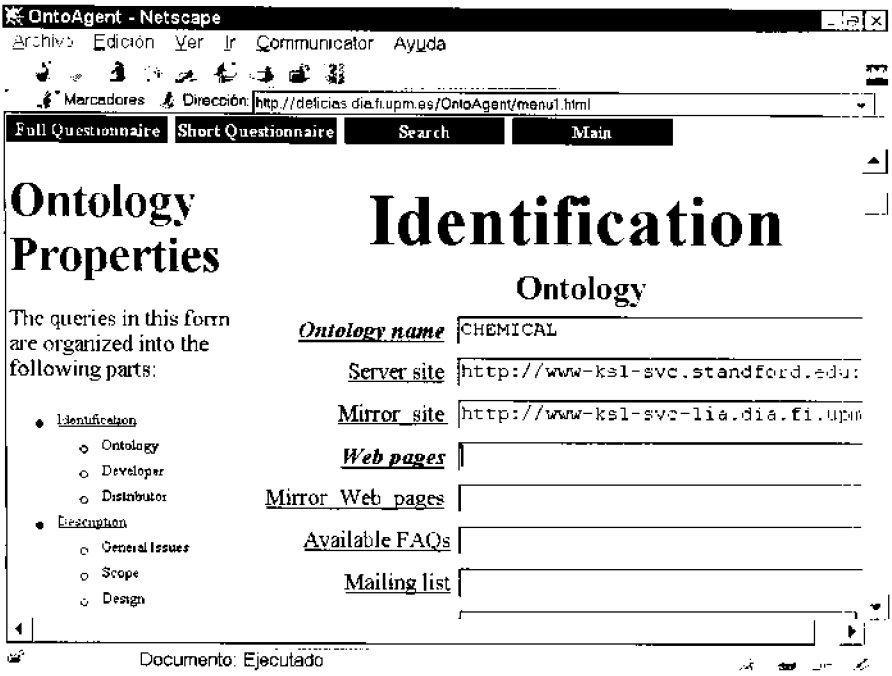

Fig. 4. HTML ontology questionnaire form.

4.2. Reference Ontology: Instances

Once the *Reference Ontology* conceptual structure has been built, a means is needed to provide its instances. The approach we took was to use a World Wide Web form based on the ontology features previously identified and presented in this paper. Its main aim is to gather information about specific ontologies and thus distribute the effort made in collecting this data from ontology developers. Part of this form (http://delicias.dia.fi.upm.es/OntoAgent/) is shown in Fig. 4.

The features on the form are classed by categories, and the categories are divided into groups. There are compulsory features that ontology developers must fill in – for example, the ontology name, the language of the ontology – while others are optional and offer a more detailed view of the ontology – for example, number of nodes at the first level. Ontology developers can choose to fill in the 70 features of the form or opt to fill in only the compulsory features. The form contains questions to obtain the values of the features of an ontology. Besides, it contains help to guide ontology developers through filling in the form. A set of possible values are also identified for some questions, so the user merely has to click on a radio button or check-box.

However, it was not enough for our purposes to merely supply the form. We needed an easy and fast way to consult the information stored in the *Reference Ontology*. This is where the *OntoAgent architecture* and $(ONTO)^2$Agent come in.

4.3. Discussion

One of the important features of the Reference Ontology is the fact that its development was divided into two phases. The first was the centralized development of its conceptual structure and the second is the distributed collection of its instances. Usually, ontologies are either centrally developed (for instance, CHEMICALS; Fernández, 1996; Fernández et al, 1999) or distributively developed (for instance, $(KA)^2$).

Another important feature is the fact that the Reference Ontology was built according to a domain ontology building methodology, which has already successfully been applied to several different domains. The inclusion of the Reference Ontology conceptual structure into the $(KA)^2$Ontology conceptual structure within ODE was simplified since we had access to it.

We believe that although knowledge represented in the Reference Ontology is more important to ontologists in particular it may also be of value to the KA community and knowledge-based system builders in general since it can spread information about existing ontologies.

The features characterizing ontologies shown in Section 3.2 provide a complete description of the characteristics to look for when ontologies are reused in a given project. However, new uses of ontologies or particular needs may require new features to be added. To face any demand from the KA community, which may arise either from ontology providers or from consumers, the procedure to follow will consist of extending the conceptual model of the Reference Ontology. In order to keep its consistency, we should follow the following steps: (1) extend and revise the conceptual model of the Reference Ontology taking into account the new knowledge; (2) add the new features to the $(ONTO)^2$Agent web form; and (3) send this new form to all ontology providers (with the previous data concerning them already filled in) so that they can update the information about their ontologies accordingly.

5. OntoAgent Architecture

Having identified the relevant features of ontologies and built the conceptual structure of the Reference Ontology using the Ontology Design Environment, the problem of accessing and updating the information about each individual ontology arises. The approach we took was to build the OntoAgent architecture: a domain-independent technology used to create and maintain ontology-based WWW brokers to retrieve knowledge from ODE-built ontologies. OntoAgent is classed in the category of agents used to easily access knowledge. Simple and complex queries can be formulated by means of several OntoAgent components. An AND/OR tree-based wizard can even be used to query the ontologies using their own vocabulary.

The approach we took to build OntoAgent is presented in Fig. 5. It consists of different modules, each of which has a major function within the system. These modules are:

A. **A World Wide Web instance collector broker**, whose main capability is to instantiate the conceptual structure of an ontology about the broker domain of expertise. This instance collector needs:

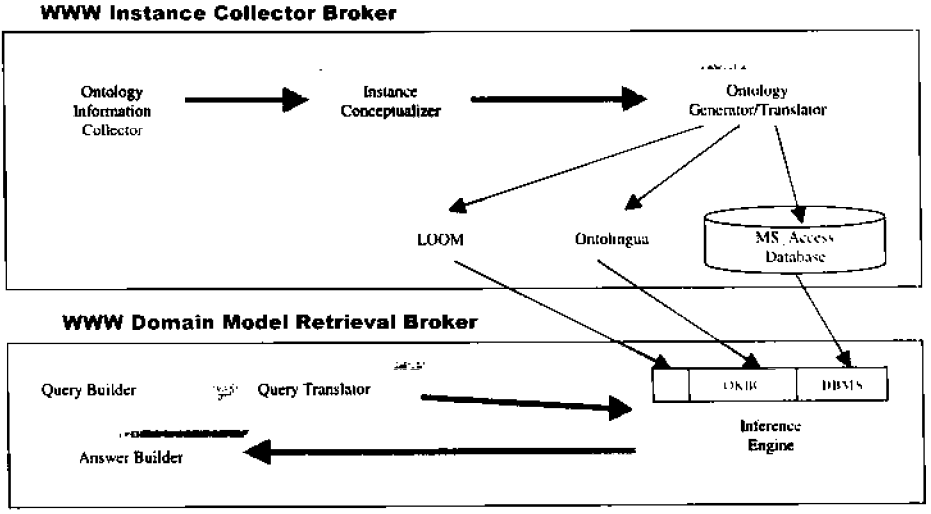

Fig. 5. The OntoAgent architecture.

A.1. *Ontology information collector*: an easy-to-use interactive WWW user interface that facilitates data input by distributed agents (both programs and humans).

A.2. *An instance conceptualizer*: for transforming the data from the WWW user interface into instances of the ontology specified at the knowledge level.

A.3. *Ontology generator/translators*: For generating or translating the instances specified at the knowledge level into several target languages used to formalize ontologies and thus allow access from heterogeneous applications.

B. **A WWW domain model retrieval broker**, whose aim is to provide help in accessing the information in an ontology warehouse and show it nicely. It is divided into:

B.1. A *query builder* to help to build queries either using the ontology vocabulary or otherwise, as well as to reformulate and refine a query given by the user. The queries will be formulated upon a set of ontologies previously selected from the ontology pool available in the architecture.

B.2. A *query translator* that transforms the user query into a query representation compatible with the format in which the ontology is implemented.

B.3. An *inference engine* that searches for the answer to the query; as shown in Fig. 5, knowledge sources can be represented in several formats.

B.4. An *answer builder* that presents the answers to the query obtained by the inference engine module to the customer in a simple and human-readable format. The answers are presented for each ontology that has been searched. Thus, one query may be answered in several domains, depending on the domains of the ontologies.

If we compare OntoAgent with its nearest counterpart (Ontobroker; Fensel et al, 1998), we can see they have several points in common. They are both distributive, joint efforts by the community (in the sense that both approaches need the community to introduce information about the broker domain expertise), they use an ontology as the source of their knowledge, they use the web to gather

information, and they have a query language for formulating queries. However, the main differences between them are as follows:

- The OntoAgent architecture uses: (1) a SQL database to formalize the ontology, (2) a WWW form and an ontology generator to store the captured knowledge, and (3) simple and complex queries based on ODE intermediate representations and AND/OR trees to retrieve information from the ontology.
- Ontobroker uses: (1) an FLogic ontology, (2) Ontocrowler for searching WWW annotated documents with ontological information, and (3) an FLogic-based syntax to formulate queries.

5.1. (ONTO)^{2}Agent

(ONTO)^{2}Agent is an instantiation of the OntoAgent architecture. This broker is specialized in querying the *Reference Ontology* to provide ontology users with a tool to quickly examine what ontologies best meet their requirements.

Using the *Reference Ontology* as a source of its knowledge, (ONTO)^{2}Agent locates and retrieves descriptions of ontologies that satisfy a given set of constraints. For example, when a knowledge engineer is looking for ontologies written in a given language applicable to a particular domain, (ONTO)^{2}Agent can help in the search, supplying the engineer with a set of ontologies that totally/partially comply with the requirements identified.

The above abstract architecture has been instantiated as follows:

A. The **WWW instance collector broker** uses:

A.1. The WWW form described in Section 4.2 to allow ontology developers to enter information about their ontologies into the *Reference Ontology*.

A.2. The data are used to fill in the instances of the concepts identified in the ontology conceptual structure described in Section 4.1, which was built using ODE, thus ensuring full compatibility with this tool.

A.3. ODE code generators allow us to implement the ontology in several target languages; however, it is important to mention at this point that (ONTO)^{2}Agent uses the relational database implementation of the ontology and not the Ontolingua implementation, because we would need to build in a GFP (Chaudhri et al, 1997) module in order to interact with Ontolingua.

B. With regard to the **WWW domain model retrieval broker**:

B.1. Two query builders have been implemented. Both builders allow users to formulate simple and complex queries. Simple queries can be made using the predefined queries present in the agent. They are based on the ODE intermediate representations presented in Section 4.1.2, and include definition of a concept, instances of a concept, etc. They are used to get answers, loaded with information, easily and quickly. One of the query methods is similar to the method employed by Yahoo or Alta Vista, so anyone used to working with these Internet search tools is unlikely to encounter any problems using the interface. Complex queries can be formulated by using a query builder wizard that works with AND/OR trees (see Fig. 6) and the vocabulary obtained from the ontologies we are querying. It allows us to build a more restrictive and detailed query. Apart from this, before the query is translated to the proper query language, it is checked semantically for inconsistencies.

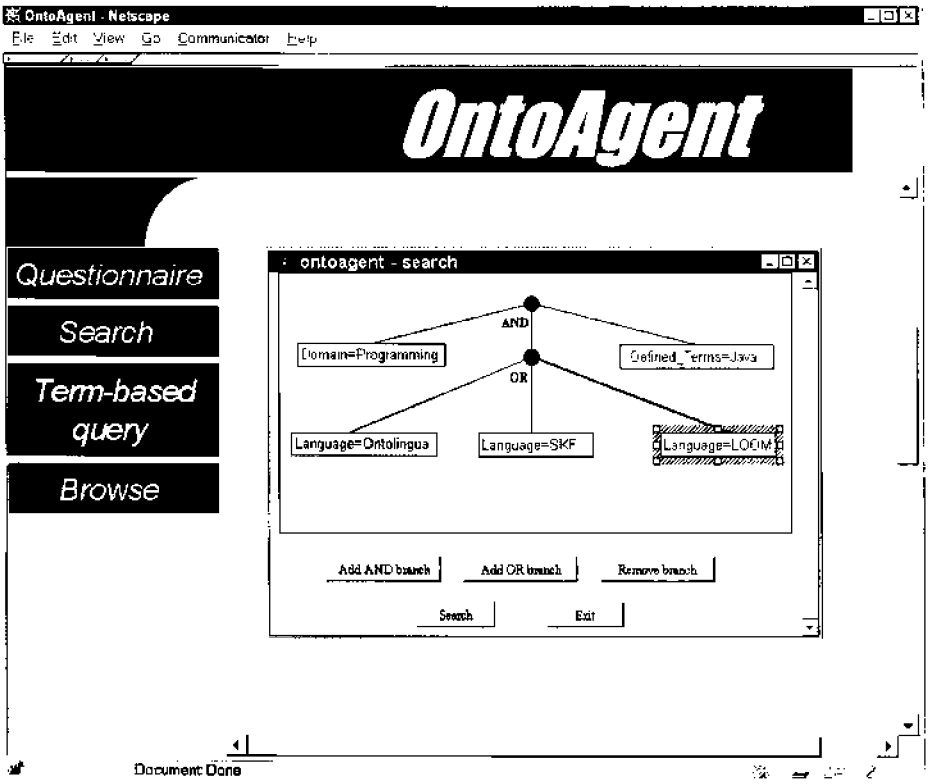

Fig. 6. (ONTO)^{2}Agent is asked to provide all the ontologies in the programming domain, written in Ontolingua, Loom or SFK, with Java as a defined term, using a query expressed by means of an AND/OR tree.

Syntactic correctness is implicit, thanks to the query building method. If it is all right, it is refined, eliminating any redundancies.

B.2. The resulting query is then translated into the SQL language in order to match the ontology implementation stored in a database. For the Ontolingua implementation of a similar agent, a GFP-capable (Chaudhri et al, 1997) builder would be required.

B.3. Once it has been sent to the OntoAgent server, the query is passed on to our *inference engine*: the MS Access search engine with a few add-ins.

B.4. The query results are returned to the client, who visualizes them within the web browser. As we use standard technology, the results can be saved in HTML format for later consultation. An example is shown in Fig. 7, where the user requested all the ontologies in the Java domain.

5.2. Chemical OntoAgent

Chemical OntoAgent is another broker to which this technology has been applied. It is a chemistry teaching broker that allows students to learn chemistry in a

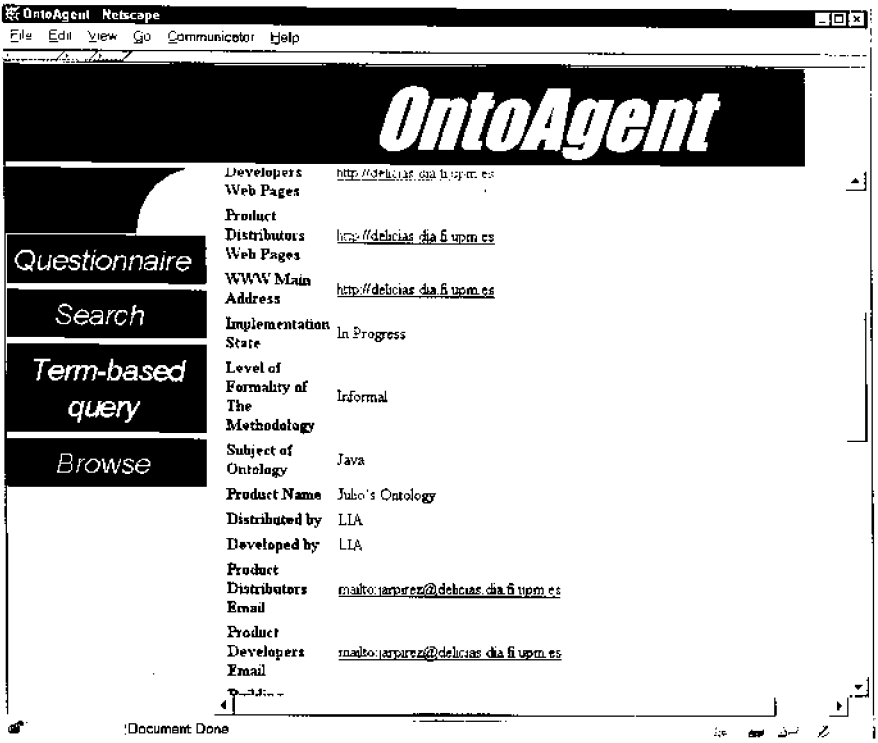

Fig. 7. HTML query result to a user request of all the ontologies in the Java domain.

very straightforward manner, providing the necessary domain knowledge and helping students to test their skills. To make the answers more understandable to students, this technology is able to interact with a system called OntoGeneration (Aguado et al, 1998). OntoGeneration is a system that uses a domain ontology (CHEMICALS; Fernández, 1996; Fernández et al, 1999) and a linguistic ontology (GUM; Bateman et al, 1995) to generate Spanish text descriptions in response to the queries in the domain of chemistry.

Chemical OntoAgent does not have the modules described for the World Wide Web instance collector broker, since the Chemicals ontology was built entirely using ODE, and needed no further dynamic updating after its completion.

6. Current Status

For the moment, the project is still in its early stages and only information about specific ontologies has been collected (there were more instances introduced but they were testing examples). Although not many instances have been introduced by the community, we have received very positive and encouraging comments.

The only complaints from the Reference Ontology instance fillers are that sometimes some of the properties are difficult to fill in (for instance, estimated price of required hardware, description of use tools) since they do not know some

of the answers or they do not know what to answer. We should stress that only ontology developers should introduce information about a given ontology since it requires a deep knowledge/understanding about it. As we stated in Section 3.2, some features cannot be used to characterize certain ontologies, and ontology builders may not know the values of some features. However, we would like to stress that there are only 22 compulsory features (those from Fig. 2 in bold type) and, usually, fillers have no difficulty in filling them.

Due to the limited amount of information collected from ontology developers, the broker can only search a limited set of ontologies. So, at this moment, we do not have feedback from real users searching ontologies for their applications using (ONTO)^{2}Agent to provide an accurate and representative evaluation from users.

7. Conclusions

In this paper we presented (ONTO)^{2}Agent, an ontology-based WWW broker to select ontologies for a given application. This application seeks to solve some important problems:

1. To solve the problem of the absence of standardized features for describing ontologies, we have presented a living and domain-independent taxonomy of 70 features to compare ontologies using the same logical organization. This framework differs from that of Hovy (1997), which was built to compare natural language processing ontologies. This framework also extends the limited number of features proposed in Fridman and Hafner (1997) for comparing well-known and representative ontologies. Further work will include a metric to weigh up features and categories that help the project manager to decide how important they are and how they are interrelated, relating them to the peculiarities and demands of their particular project.
2. To solve the problem of the dispersion of ontologies over several servers and the absence of common formats for representing relevant information about ontologies using the same logical organization, we built a living *Reference Ontology* (a domain ontology about ontologies) that gathers, describes using the same logical organization and has links to existing ontologies. We built the conceptual structure ontology at the knowledge level using the METHONTOLOGY framework and the Ontology Design Environment. Knowledge about specific ontologies that act as instances in the Reference Ontology are entered by ontology developers using a WWW form based on the previously identified features.
3. To solve the problem of searching for and locating candidate ontologies over several servers, we built (ONTO)^{2}Agent, an ontology-based WWW broker that retrieves the ontologies that satisfy a given set of constraints using the knowledge formalized in the Reference Ontology. (ONTO)^{2}Agent is an instantiation of the OntoAgent architecture. This is a domain-independent technology used to create and maintain ontology-based WWW brokers to retrieve its knowledge from ODE-built ontologies.

We hope that (ONTO)^{2}Agent and the Reference Ontology will facilitate the search for ontologies to be used in other applications.

Acknowledgements. We would like to thank Mariano Fernandez and Juanma Garcia for their help in using ODE. We would also like to thank the anonymous reviewers of the ECAI98 workshop on Applications of Ontologies and Problem Solving Methods and KAIS journal anonymous reviewers for their comments. This work was partially supported by JNICT grant PRAXIS XXI/BD/11202/97 (Sub-Programa Ciência e Tecnologia do Segundo Quadro Comunitário de Apoio).

References

Aguado G, Bateman J, Bañón A, Bernardos S, Fernández M, Gómez-Pérez A, Nieto E, Olalla A, Plaza R, Sanchez A (1998) ONTOGENERATION: reusing domain and linguistic ontologies for Spanish. In Workshop on applications of ontologies and PSMs, Brighton, UK, 1–10 August 1998

Arpírez JC, Gómez-Pérez A, Lozano-Tello A, Pinto S (1998) (ONTO)2Agent: an ontology-based WWW broker to select ontologies. In Workshop on applications of ontologies and problem-solving methods, ECAI '98, Brighton, UK, pp 16–24

Basili V, Briand L, Thomas W (1994) Domain analysis for the reuse of software development experiences. In Proceedings of the 19th annual software engineering workshop, NASA/GSFC, Greenbelt, MD

Bateman JA, Magnini B, Rinaldi F (1994) The generalized Italian, German, English upper model. In Proceedings of the ECAI '94 workshop on comparison of implemented ontologies, Amsterdam

Bateman JA, Magnini B, Fabris G (1995) The generalized upper model knowledge base: organization and use. In Mars, N (ed), Towards very large knowledge bases. IOS Press, Amsterdam, pp 60–72

Benjamins R, Fensel D (1998) Problem solving methods. International Journal of Human Computer Studies 49: 305–313

Benjamins R, Decker S, Fensel D, Motta E, Plaza E, Schreiber G, Studer R, Wielinga B (1998) IBROW3 an intelligent brokering service for knowledge-component reuse on the World-Wide Web. In Workshop on applications of ontologies and problem-solving methods, ECAI '98, Brighton, UK, pp 25–30

Benjamins R, Fensel D, Decker S, Gómez-Pérez A (1999) $(KA)^2$: building ontologies for the Internet: a mid term report. International Journal of Human Computer Studies 51: 687–712

Blázquez M, Fernández M, García-Pinar JM, Gómez-Pérez A (1998) Building ontologies at the knowledge level using the ontology design environment. In Knowledge acquisition workshop, KAW '98, Banff, Alberta, Canada

Borst P (1997) Construction of engineering ontologies for knowledge sharing and reuse. PhD dissertation, University of Twente

Borst P, Benjamins J, Wielinga B, Akkermans H (1996) An application of ontology construction. In Workshop on ontological engineering, ECAI '96, Budapest, pp 5–16

Bylander T, Chandrasekaran B (1988) Generic tasks in knowledge-based reasoning: the right level of abstraction for knowledge acquisition. In Gaines BR, Boose JH (eds). Knowledge acquisition based systems, vol 1. Academic Press, London, pp 65–77

Chandrasekaran B, Josephson JR, Benjamins VR (1998) Ontology of tasks and methods. In Workshop on applications of ontologies and PSMs, Brighton, UK, August, pp 31–43

Chaudhri VK, Farquhar A, Fikes R, Karp PD, Rice JP (1997) The generic frame protocol 2.0, 21 July 1997. Stanford University, Stanford, USA

Dahlgren K (1988) Naive semantics for natural language understanding. Kluwer, Boston, MA

Farquhar A, Fikes R, Pratt W, Rice J (1995) Collaborative ontology construction for information integration. Technical report KSL-95-10, Knowledge Systems Laboratory, Stanford University, CA

Farquhar A, Fikes R, Rice J (1996) The Ontolingua server: a tool for collaborative ontology construction. In Proceedings of the 10th knowledge acquisition for knowledge-based systems workshop, vol 19, Banff, Alberta, Canada, pp 1–44

Fensel D, Decker S, Erdman M, Studer R (1998) Ontobroker: the very high idea. In Proceedings of the 11th international Flairs conference (FLAIRS '98), Sanibal Island, FL, May 1998

Fernández M (1996) CHEMICALS: ontología de elementos químicos. Proyecto fin de carrera. Facultad de Informática, Universidad Politécnica de Madrid, December 1996

Fernández M, Gómez-Pérez A, Juristo N (1997) METHONTOLOGY: from ontological art toward ontological engineering. In Spring symposium series on ontological engineering, AAAI '97, Stanford, CA, March 1997

Fernández M, Gómez-Pérez A, Pazos A, Pazos J (1999) Building a chemical ontology using METHONTOLOGY and the ontology design environment. IEEE Intelligent Systems 14(1): 37–46

Fikes R, Farquhar A (1997) Large-scale repositories of highly expressive reusable knowledge. Knowledge Systems Laboratory, KSL-97-02, April 1997. Stanford University, Stanford, USA
Foundation for Intelligent Physical Agents (1998) FIPA '98 specification: ontology service. FIPA, October 1998
Fridman N, Hafner C (1997) The state of the art in ontology design. AI Magazine Fall: 53–74
Genesereth M, Fikes R (1992) Knowledge interchange format. Technical report, Logic-92-1. Computer Science Department, Stanford University, Stanford, CA
Gómez-Pérez A (1996) Towards a framework to verify knowledge sharing technology. Expert Systems with Applications 11(4): 519–529
Gómez-Pérez A (1998) Knowledge sharing and reuse. In Liebowitz J (ed). Handbook of applied expert systems. CRC Press, Boca Raton, FL
Gómez-Pérez A, Benjamins VR (1998) Preface: applications of ontologies and problem solving methods. Preface of the Workshop on applications of ontologies and problem-solving methods, ECAI '98, Brighton, UK, pp i–iv
Gruber G (1993) Translation approach to portable ontology specifications. Knowledge Acquisition 5: technical report
Gruber T (1993) Toward principles for the design of ontologies used for knowledge sharing. Technical report KSL-93-04, Knowledge Systems Laboratory, Stanford University, Stanford, CA
Gruber T, Olsen R (1994) An ontology for engineering mathematics. Technical report KSL-94-18, Knowledge Systems Laboratory, Stanford University, Stanford, CA
Gruninger M, Fox M (1995) Methodology for the design and evaluation of ontologies. In Proceedings of the IJCAI '95 workshop on basic ontological issues in knowledge sharing. Montreal, Canada
Hovy E (1997) What would it mean to measure an ontology? Unpublished
Humphreys B, Lindberg D (1993) UMLS project: making the conceptual connection between users and the information they need. Bulletin of the Medical Library Association 81(2): 170–7
Kan SK (1995) Metrics and models in software quality engineering. Addison-Wesley, Reading, MA
Karp PD (1993) A qualitative biochemistry and its application to the regulation of the tryptophan operon. In Hunter L (ed). Artificial intelligence and molecular biology. AAAI Press/MIT Press, Cambridge, MA, pp 289–325
Khairuddin H, Key E (1995) A software reusability attributes model. International Journal of Computer Applications in Technology 8(1–2): 69–77
Kifer M, Lausen G, Wu J (1995) Logical foundations of object-oriented and frame-based languages. Journal of the ACM 42: 741–843
Lenat DB (1990) CYC: toward programs with common sense. Communications of the ACM 33(8): 30–49
MacGregor R (1991) Inside the Loom classifier. SIGART Bulletin 2(3): 70–76
Miller GA (1990) WordNet: an on-line lexical database. International Journal of Lexicography 3(4): 235–312
Mizoguchi R, Vanwelkenhuysen J, Ikeda M (1995) Task ontology for reuse of problem solving knowledge. In Mars, N (ed), Towards very large knowledge bases: knowledge building and knowledge sharing. IOS Press, Amsterdam, pp 46–59
Pressman R (1997) Software engineering: a practitioner's approach. McGraw-Hill
Slagle J, Wick M (1988) A method for evaluating candidate. AI Magazine 88: 44–53
Sowa JF (1997) Knowledge representation: logical, philosophical, and a computational foundations. PWS, Boston, MA
Studer R, Benjamins R, Fensel D (1998) Knowledge engineering: principles and methods. Data and Knowledge Engineering 25: 161–197
Swartout B, Patil R, Knight K, Russ T (1997) Towards distributed use of large-scale ontologies, AAAI '97 Spring symposium series on ontological engineering, Stanford University, Stanford, USA
Tennison J, Shadbolt N (1998) APECKS: a tool to support living ontologies. In Proceedings of the 11th Banff knowledge acquisition for knowledge-based systems workshop, Banff, Alberta, Canada
Uschold M (1998) Where are the killer apps? In Workshop on applications of ontologies and problem-solving methods, ECAI '98, Brighton, UK, pp 107–111
Uschold M, Grüninger M (1996) ONTOLOGIES: principles, methods and applications. Knowledge Engineering Review 11(2): 93–155
Van der Vet PE, Speel P, Mars N (1994) The Plinius ontology of ceramic materials. In Proceedings of the ECAI '94 workshop on comparison of implemented ontologies, Amsterdam
van Heist G, Schreiber A, Wielinga BJ (1997) Using explicit ontologies in KBS development. International Journal of Human–Computer Studies 45: 183–292