Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Data & Knowledge Engineering 46 (2003) 41–64

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

Methodologies, tools and languages for building ontologies. Where is their meeting point?

Oscar Corcho [1], Mariano Fernández-López [2], Asunción Gómez-Pérez *

Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo s/n, Boadilla del Monte, Madrid 28660, Spain

Received 28 November 2001; received in revised form 21 August 2002; accepted 30 October 2002

Abstract

In this paper we review and compare the main methodologies, tools and languages for building ontologies that have been reported in the literature, as well as the main relationships among them. Ontology technology is nowadays mature enough: many methodologies, tools and languages are already available. The future work in this field should be driven towards the creation of a common integrated workbench for ontology developers to facilitate ontology development, exchange, evaluation, evolution and management, to provide methodological support for these tasks, and translations to and from different ontology languages. This workbench should not be created from scratch, but instead integrating the technology components that are currently available.

Keywords: Ontology; Ontology methodology; Ontology language; Ontology tool

1. Introduction

In the last decade, the word ''ontology'' has become a fashionable word inside the Knowledge Engineering Community. We have seen many definitions about what an ontology is and we have also seen how such definitions have changed and evolved over the time.

* Corresponding author. Tel.: +34-913367439; fax: +34-913524819.
E-mail addresses: ocorcho@fi.upm.es (O. Corcho), mfernandez@fi.upm.es (M. Fernández-López), asun@fi.upm.es (A. Gómez-Pérez).
[1] Tel.: +34-913366604.
[2] Tel.: +34-913366605.

0169-023X/03/$ - see front matter
doi:10.1016/S0169-023X(02)00195-7

When a new ontology is going to be built, several basic questions arise related to the methodologies, tools and languages to be used in its development process:

- Which methods and methodologies can I use for building ontologies, either from scratch, or reusing other ontologies already available on ontology servers? Which activities are performed when building ontologies with a methodology? Does any methodology support building ontologies cooperatively? Which is the life-cycle of an ontology that is developed with a specific methodology?
- Which tool(s) give/s support to the ontology development process? How are the ontologies stored (in databases or files)? Does the tool have an inference engine? Do tools have translators for different ontology languages or formats? What is the quality of the translations? How can applications interoperate with ontology servers and/or use the ontologies that we have developed?
- Which language(s) should I use to implement my ontology? What expressiveness has an ontology language? What are the inference mechanisms attached to an ontology language? Does any ontology development tool support the language? Is the language chosen appropriate for exchanging information between different applications? Does the language ease the integration of the ontology in an application? Is the language compatible with other languages used to represent knowledge and information on the Web? Does my application require having implemented the ontology in different languages? Does my application require integrating existing ontologies that were already implemented in different languages? Are there translators that transform the ontology implemented in a source language in a target language? and finally, how do such translators minimize the loss of knowledge in the translation process?

Along this paper, we will present the main characteristics of methodologies, tools and languages, which can help practitioners and researchers in this field to obtain answers to the previous questions. That is, we will provide guidelines that help selecting the most appropriate methodologies, tools and languages for building an ontology in a specific domain. First, we will briefly comment on some definitions of the term ontology.

2. What is an ontology?

The word ontology was taken from Philosophy, where it means a systematic explanation of being. In the last decade, the word ontology became a relevant word for the Knowledge Engineering community. We have read many definitions about what an ontology is and have also observed how such definitions have changed and evolved over the time. In this section, we will review these definitions and explain the relationships between them.

One of the first definitions was given by Neches and colleagues [57], who defined an ontology as follows: ''an ontology defines the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary''. This descriptive definition tells what to do in order to build an ontology, and gives us some vague guidelines: the definition identifies basic terms and relations between terms, identifies rules to combine terms, and provides the definitions of such terms and relations. Note that, according to Neches' definition, an ontology includes not only the terms that are explicitly defined in it, but also

the knowledge that can be inferred from it. A few years later, Gruber [31] defined an ontology as "an explicit specification of a conceptualization". This definition became the most quoted in literature and by the ontology community. Based on Gruber's definition, many definitions of what an ontology is were proposed. Borst [6] modified slightly Gruber's definition: "Ontologies are defined as a formal specification of a shared conceptualization". Gruber's and Borst's definitions have been merged and explained by Studer and colleagues [64] as follows: "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 of some individual, but accepted by a group". In 1995, Guarino and colleagues [34] collected and analyzed seven definitions of ontologies and provided their corresponding syntactic and semantic interpretations. In that paper, the authors proposed to consider an ontology as "a logical theory which gives an explicit, partial account of a conceptualization", where conceptualization is basically an idea of the world that a person or a group of people can have. Although on the surface the notion of conceptualization is quite similar to Studer and colleagues' [64] notion, we can say that Guarino and colleagues [34] went a step forward because they established how to build the ontology by making a logical theory. Hence, strictly speaking, this definition would be only applicable to ontologies developed in logic.

There also exists another group of definitions based on the process followed to build the ontology. These definitions also include some highlights about the relationship between ontologies and knowledge bases. For example, the definition given by Bernaras and colleagues [4] in the framework of the KACTUS project [61]: "it [an ontology] provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base". Note that this definition proposes 'extracting' the ontology from a knowledge base (KB), which reflects the approach the authors use to build ontologies. In this approach, the ontology is built, following a bottom-up strategy, on the basis of an application KB, by means of an abstraction process. As more applications are built, the ontology becomes more general, and, therefore, moves further away from what would be a KB.

Another strategy for building ontologies would be to reuse large ontologies like SENSUS [66] (with more than 70,000 concepts) to build domain specific ontologies and KBs: "an ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base". According to this definition, the same ontology can be used for building several KBs, which would share the same skeleton. Extensions of the skeleton should be possible at the low level by adding domain-specific subconcepts, or at the high level by adding intermediate or upper level concepts that cover new areas. If systems are built with the same ontology, they share a common underlying structure, therefore, merging and sharing their KBs and inference mechanisms will become easier.

Sometimes, the notion of ontology is diluted, in the sense that taxonomies are considered full ontologies [64]. For instance, UNSPSC [3], e-cl@ss [4], and RosettaNet [5], which are standards on the

[3] http://www.unspsc.org/.
[4] http://www.eclass.org/.
[5] http://www.rosettanet.org/.

e-commerce domain, and the Yahoo! Directory, which is a taxonomy for searching the web, are also considered ontologies [51] because they provide a consensual conceptualization of a given domain. The ontology community distinguishes ontologies that are mainly taxonomies from ontologies that model the domain in a deeper way and provide *more restrictions* on domain semantics. The community calls them *lightweight and heavyweight ontologies* respectively. On the one hand, lightweight ontologies include concepts, concept taxonomies, relationships between concepts and properties that describe concepts. On the other hand, heavyweight ontologies add axioms and constraints to lightweight ontologies.

Since ontologies are widely used for different purposes (natural language processing, knowledge management, e-commerce, intelligent integration information, the semantic web, etc.) in different communities (i.e., knowledge engineering, databases and software engineering), Uschold and Jasper [69] provided a new definition of the word ontology to popularize it in other disciplines. Note that the database community, as well as the object-oriented community, also builds domain models using concepts, relations, properties, etc., but most of the times both communities impose less semantic constraints than those imposed in heavyweight ontologies. Uschold and Jasper defined an ontology as follows: "An ontology may take a variety of forms, but it will necessarily include a vocabulary of terms and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms."

In this section we have collected the most relevant definitions, although there are other definitions of the word "ontology" in literature. However, we can say that there is consensus among the ontology community and so there is not confusion about its usage. Different definitions provide different and complementary points of view of the same reality. Some authors provide definitions that are independent of the processes followed to build the ontology and also independent of its use in applications, while other definitions are influenced by the ontology development process. As a main conclusion to this section, we can say that ontologies aim to capture *consensual* knowledge in a generic and formal way, and that they may be reused and shared across applications (software) and by groups of people. Ontologies are usually built cooperatively by a group of people in different locations.

3. Methods and methodologies for building ontologies

Basically, a series of approaches have been reported for developing ontologies. In 1990, Lenat and Guha published the general steps [52] and some interesting points about the Cyc development. Some years later, in 1995, on the basis of the experience gathered in developing the Enterprise Ontology [70] and the TOVE (TOronto Virtual Enterprise) project ontology [33] (both in the domain of enterprise modelling), the first guidelines were proposed and later refined in [67,68]. At the 12th European Conference for Artificial Intelligence (ECAI'96), Bernaras et al. [4] presented a method used to build an ontology in the domain of electrical networks as part of the Esprit KACTUS [61] project. The methodology METHONTOLOGY [28] appeared at the same time and was extended in later papers [20,22,26]. In 1997, a new method was proposed for building ontologies based on the SENSUS ontology [66]. Some years later, the On-To-Knowledge methodology appeared as a result of the project with the same name [62]. However, all these

methods and methodologies do not consider collaborative and distributed construction of ontologies. The only method that includes a proposal for collaborative construction is CO4 [17]. This method includes a protocol for agreeing new pieces of knowledge with the rest of the knowledge architecture, which has been previously agreed. A comparative and detailed study of these methods and methodologies can be found at [21].

All the previous methods and methodologies were proposed for building ontologies. However, many other methods and methodologies have been proposed for other tasks, such as ontology reengineering [29], ontology learning [2,45], ontology evaluation [25,27,35,36,41,42], ontology evolution [47,48,58,63], ontology merging [59], etc. In this paper, we will only focus on methodologies for building ontologies.

The method used to build the *Cyc* KB [52] consists of three phases. The first phase consists of the manual codification of articles and pieces of knowledge, in which common sense knowledge that is implicit in different sources is extracted by hand. The second and third phases consist of acquiring new common sense knowledge using natural language or machine learning tools. The difference between them is that in the second phase this common sense knowledge acquisition is aided by tools, but mainly performed by humans, while in the third phase the acquisition is mainly performed by tools.

The *Uschold and King's* method [70] proposes four activities: (1) to identify the purpose of the ontology, (2) to build it, (3) to evaluate it, and (4) to document it. During the building activity, the authors propose capturing knowledge, coding it and integrating other ontologies inside the current one. The authors also propose three strategies for identifying the main concepts in the ontology: a top-down approach, in which the most abstract concepts are identified first, and then, specialized into more specific concepts; a bottom-up approach, in which the most specific concepts are identified first and then generalized into more abstract concepts; and a middle-out approach, in which the most important concepts are identified first and then generalized and specialized into other concepts.

Grüninger and Fox [33] propose a methodology that is inspired on the development of knowledge-based systems using first order logic. They propose first to identify intuitively the main scenarios (possible applications in which the ontology will be used). Then, a set of natural language questions, called competency questions, are used to determine the scope of the ontology. These questions and their answers are used both to extract the main concepts and their properties, relations and axioms on the ontology. Such ontology components are formally expressed in first-order logic. Therefore, this is a very formal method that takes advantage of the robustness of classic logic. It can be used as a guide to transform informal scenarios in computable models.

In the method proposed at the KACTUS project [4] the ontology is built on the basis of an application KB, by means of a process of abstraction (that is, following a bottom-up strategy). The more applications are built, the more general the ontology becomes; hence, the further the ontology moves away from a KB. In other words, they propose to start building a KB for a specific application. Later, when a new KB in a similar domain is needed, they propose to generalize the first KB into an ontology and adapt it for both applications. Applying this method recursively, the ontology would represent the consensual knowledge needed in all the applications.

The method based on *Sensus* [66] is a top-down approach for deriving domain specific ontologies from huge ontologies. The authors propose to identify a set of "seed" terms that are relevant to a particular domain. These terms are linked manually to a broad-coverage ontology (in this case, the Sensus ontology, which contains more than 70,000 concepts). Then, all the concepts in the path

from the seed terms to the root of SENSUS are included. If a term that could be relevant in the domain has not yet appeared, it is manually added, and the previous step is performed again, until no term is missing. Finally, for those nodes that have a large number of paths through them, the entire subtree under the node is added, based on the idea that if many of the nodes in a subtree have been found to be relevant, then, the other nodes in the subtree are likely to be relevant as well. Consequently, this approach promotes sharability of knowledge, since the same base ontology is used to develop ontologies in particular domains.

METHONTOLOGY [20] is a methodology, created in the Artificial Intelligence Lab from the Technical University of Madrid (UPM), for building ontologies either from scratch, reusing other ontologies as they are, or by a process of reengineering them. The METHONTOLOGY framework enables the construction of ontologies at the knowledge level. It includes: the identification of the ontology development process, a life cycle based on evolving prototypes (shown in Fig. 1), and particular techniques to carry out each activity. The *ontology development process* identifies which tasks should be performed when building ontologies (scheduling, control, quality assurance, specification, knowledge acquisition, conceptualization, integration, formalization, implementation, evaluation, maintenance, documentation and configuration management). The *life cycle* identifies the stages through which the ontology passes during its lifetime, as well as the interdependencies with the life cycle of other ontologies [19]. Finally, the methodology specifies the techniques used in each activity, the products that each activity outputs and how they have to be evaluated. The main phase in the ontology development process using the METHONTOLOGY approach is the conceptualization phase. Tools such as WebODE [1,12] and ODE [5] provide support to METHONTOLOGY. However, other tools can be also used to develop ontologies following this methodology.

The On-To-Knowledge methodology [62] includes the identification of goals that should be achieved by knowledge management tools and is based on an analysis of usage scenarios. The steps proposed by the methodology are: *kick-off*, where ontology requirements are captured and

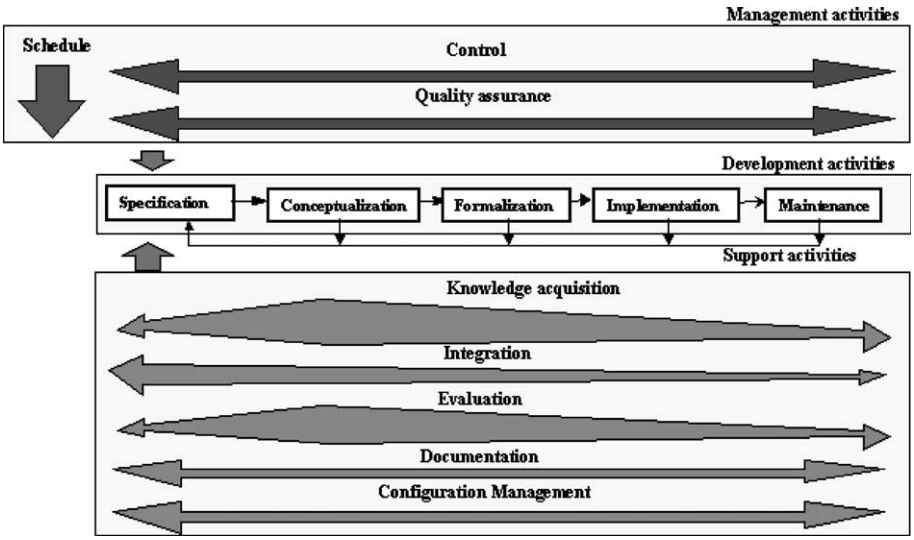

Fig. 1. Ontology life-cycle in METHONTOLOGY.

specified, competency questions are identified, potentially reusable ontologies are studied and a first draft version of the ontology is built; *refinement*, where a mature and application-oriented ontology is produced; *evaluation*, where the requirements and competency questions are checked, and the ontology is tested in the application environment; and *ontology maintenance*.

Finally, *CO4* [17] is a protocol to reach consensus between several KBs, which are organized in a tree. The leaves are called user KBs, and the intermediate nodes, group KBs. The user KBs do not need have consensual knowledge. In each intermediate node, there is knowledge consensuated among all its children and siblings. Knowledge consensus is achieved by the exchange of messages between users.

In this section, we have presented several methodologies and methods for building ontologies, each one following different approaches. For instance, if we compare the KACTUS and the Sensus methods, in the first one the ontology is built by means of an abstraction process from an initial knowledge base, while in the second one the ontology skeleton is automatically generated from a huge ontology. The other methods and methodologies can be used for building ontologies either from scratch or reusing other ontologies.

These approaches can be also compared taking into account the degree of dependency of the developed ontology and its final application. In this sense, we can conclude that: (a) the method used at the KACTUS project and the On-To-Knowledge methodology are application dependent, since the ontology is built on the basis of a given application; (b) the Grüninger and Fox methodology and the method based on Sensus are semi application-dependent; and (c) the Cyc and the Uschold and King methods, and the methodology METHONTOLOGY are application-independent, since the ontology development process is totally independent of the uses of the ontology.

According to the previous analysis and the work reported at [21], we can conclude that:

(a) None of the approaches presented is fully mature if we compare them with software engineering and knowledge engineering methodologies. As summarized in Table 1, many key activities are not proposed by most of them. The most mature approach is METHONTOLOGY, which has been recommended by FIPA for the ontology construction task.
(b) Current proposals are not unified: each group applies its own approach. Consequently, great effort is required for creating a consensuated methodology for ontology construction. Collaboration between different groups to unify their approaches seems the most reasonable way to achieve it.

Table 1 summarizes which activities are proposed in each of the approaches that have been presented, hence answering to several questions presented in the Section 1 (specifically, the first, third and fourth questions).

4. Ontology development tools

In the last years, the number of environments and tools for building ontologies has grown exponentially. These tools are aimed at providing support for the ontology development process and for the subsequent ontology usage. In this section, the most relevant ones are presented.

Table 1
A comparison of methodologies for building ontologies

Feature			Cyc	Usdhold and King	Grüninger and Fox	KACTUS	METHONTOLOGY	SENSUS	On-To-Knowledge
Project management processes	Project initiation		Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Proposed
	Project monitoring and control		Not proposed	Not proposed	Not proposed	Not proposed	Proposed	Not proposed	Proposed
	Ontology quality management		Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Proposed
Ontology development-oriented processes	Pre-development processes	Concept exploration	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Proposed
		System allocation	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed
	Development processes	Requirements	Not proposed	Proposed	Proposed	Proposed	Described in detail	Proposed	Proposed
		Design	Not proposed	Not proposed	Described	Described	Described in detail	Not proposed	Proposed
		Implementation	Proposed	Proposed	Described	Proposed	Described in detail	Described	Proposed
	Post-development processes	Installation	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed
		Operation	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed
		Support	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed
		Maintenance	Not proposed	Not proposed	Not proposed	Not proposed	Proposed	Not proposed	Proposed
		Retirement	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed
Integral processes	Knowledge acquisition		Proposed	Proposed	Proposed	Not proposed	Described in detail	Not proposed	Proposed
	Verification and validation		Not proposed	Proposed	Proposed	Not proposed	Described in detail	Not proposed	Proposed
	Ontology configuration management		Not proposed	Not proposed	Not proposed	Not proposed	Described in detail	Not proposed	Proposed
	Documentation		Proposed	Proposed	Proposed	Not proposed	Described in detail	Not proposed	Proposed
	Training		Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed	Not proposed

The *Ontolingua Server* [18] was the first ontology tool created. It was developed in the Knowledge Systems Laboratory (KSL) at Stanford University. The Ontolingua Server appeared at the beginning of the 1990s, and was built to ease the development of Ontolingua ontologies with a form-based web application. Initially, the main module inside the Ontolingua Server was the ontology editor, then other modules were included in the environment, such as a Webster, an equation solver, an OKBC (Open Knowledge Based Connectivity) server, Chimaera (an ontology merging tool) [55], etc. The ontology editor also provides translators to languages, such as Loom, Prolog, CORBA's IDL, CLIPS, etc. Remote editors can browse and edit ontologies, and remote or local applications can access any of the ontologies in the ontology library with the OKBC protocol [10].

At the same time, *Ontosaurus* [66] was developed by the Information Sciences Institute (ISI) at the University of South California. OntoSaurus consists of two modules: an ontology server, which uses Loom as its knowledge representation system, and a web browser for Loom ontologies. Translators from Loom to Ontolingua, KIF, KRSS and C++ are available. OntoSaurus ontologies can be also accessed with the OKBC protocol.

In 1997, the Knowledge Media Institute (KMI) at the Open University developed *Tadzebao* and *WebOnto* [15]. WebOnto is an ontology editor for OCML ontologies. Its main advantage over other available tools is that it supports editing ontologies collaboratively, allowing synchronous and asynchronous discussions about the ontologies being developed.

The main similarity among the aforementioned environments is that all of them have a strong relationship with a specific language (Ontolingua, LOOM and OCML, respectively). Actually, they were created to allow browsing and editing easily ontologies in those languages. Furthermore, they were strictly oriented to research activities and most of them were built as isolated tools that did not provide many extensibility facilities.

In the last years, a new generation of ontology-engineering environments have been developed. The design criteria of these environments is much more ambitious than those of the tools mentioned. They have been created to integrate ontology technology in actual information systems. As a matter of fact, they are built as robust integrated environments or suites that provide technological support to most of the ontology lifecycle activities. They have extensible, component-based architectures, where new modules can easily be added to provide more functionality to the environment. Besides, the knowledge models underlying these environments are language independent. Among these environments, we can cite Protégé 2000, WebODE and OntoEdit.

Protégé 2000 [60] has been developed by the Stanford Medical Informatics (SMI) at Stanford University, and is the latest version of the Protégé line of tools. It is an open source, standalone application with an extensible architecture. The core of this environment is the ontology editor, and it holds a library of plugins that add more functionality to the environment. Currently, plugins are available for ontology language importation/exportation (FLogic, Jess, OIL, XML, Prolog), OKBC access, constraints creation and execution (PAL), ontology merge (PROMPT [59]), etc.

WebODE [1] is the successor of ODE (Ontology Design Environment) [5], and has been developed in the Artificial Intelligence Lab from the Technical University of Madrid (UPM). It is also an ontology-engineering suite created with an extensible architecture. WebODE is not used as a standalone application, but as a Web server with a Web interface. The core of this environment is the ontology access service, which is used by all the services and applications plugged into the server, especially by the WebODE's Ontology Editor. There are several services for ontology

language importation/exportation (XML, RDF(S), OIL, DAML + OIL, CARIN, FLogic, Jess, Prolog), axiom edition with WebODE Axiom Builder (WAB) [12], ontology documentation, ontology evaluation and ontology merge. WebODE's ontologies are stored in a relational database. Finally, WebODE covers and gives support to most of the activities involved in the ontology development process proposed by METHONTOLOGY, although this does not prevent it from being used with other methodologies or without following any methodology.

OntoEdit [65] has been developed by AIFB in Karlsruhe University. It is similar to the previous tools: it is an extensible and flexible environment, based on a plugin architecture, which provides functionality to browse and edit ontologies. It includes plugins that are in charge of inferring using Ontobroker [14], of exporting and importing ontologies in different formats (FLogic, XML, RDF(S), DAML + OIL), etc. Two versions of OntoEdit are available: OntoEdit Free and OntoEdit Professional. Recently, the KAON (Karlsruhe Ontology) tool suite has been developed as the successor of OntoEdit.

Finally, with the huge emergence of the Semantic Web, tools for the development of DAML + OIL and RDF(S) ontologies have proliferated. In fact, the previous suites (Protégé 2000, WebODE and OntoEdit) allow importing and exporting DAML + OIL and RDF(S) ontologies. There are also several isolated tools that create DAML + OIL ontologies from different perspectives; the most representative are: OILEd (a DL based tool), and DUET (a UML-based plugin for Rational Rose).

OILEd [3] was initially developed as an ontology editor for OIL ontologies, in the context of the European IST On-To-Knowledge project. The University of Manchester, the Free University of Amsterdam and Interprice GmbH participated in this development. However, OILEd has evolved and now is an editor of DAML + OIL ontologies. OILEd users can connect to the FaCT [39] inference engine, which provides consistency checking and automatic concept classification features. OILEd also provides several documentation options (HTML, graphical visualization of ontologies, etc.).

DUET [49] is being developed by AT&T Government Solutions Advanced Systems Group. It offers a UML visualization and authoring environment for DAML + OIL, which is integrated as a plugin in the Rational Rose suite. Core DAML + OIL concepts are being mapped into UML through a UML profile for DAML + OIL. This tool is not intended for knowledge engineers but for database designers and systems engineers, who can model their ontologies with UML and then translate them into DAML + OIL, which can be applied to the software systems they are developing. This is not the only tool that is integrated as a plugin in the Rational Rose suite; for instance, the Medius Visual Ontology Modeller (*VOM*) [44] has been also created similarly.

Many other ontology-related tools with other purposes exist nowadays: there are tools specialized in ontology merge (Chimaera [55], Protégé-PROMPT [59]), ontology translation between languages (Ontomorph [9]), ontology-based web page annotation (COHSE, OntoMat, SHOE Knowledge Annotator), ontology evaluation (OntoAnalyser, ONE-T, ODEClean), RDF query engines (RDFSuite, Sesame, Inkling, Jena, etc.), etc. However, in this study we have focused only on ontology development tools.

Below we present some conclusions of our work with the tools presented above, whose results are also summarized in Table 2. We have compared all these tools with respect to the same eval-

uation framework, extending the one used in [11,16]. This comparative study has been performed in the context of the SIG on Enterprise-Standard Ontology Environments of the OntoWeb thematic network, and is published in its deliverable D1.3 (''A survey on ontology tools'') [30].

From the *KR paradigm* point of view, there are two families of tools. OILEd and OntoSaurus, which are description-logic based tools, and the rest of tools, which allow representing knowledge following a hybrid approach based on frames and first order logic. *Expressiveness of the underlying tool knowledge model* is also important. All the tools allow representing classes, partitions, relations, attributes, instances and axioms. However, only the Ontolingua Server, Ontosaurus and Protégé 2000 provide flexible modelling components like metaclasses. Before selecting a tool for developing an ontology, it is also important to know the *inference services* attached to the tool, which includes: constraint and consistency checking mechanisms, type of inheritance (single, multiple, monotonic, non-monotonic), automatic classifications, exception handling and execution of procedures. We can say that the Ontolingua Server and Protégé 2000 do not have an inference engine. OILEd performs inferences using FACT inference engine [39], OntoEdit uses Ontobroker [14], Ontosaurus uses the Loom classifier [54], WebODE uses Ciao Prolog [37] and WebOnto uses OCML [56]. Besides, Ontosaurus and WebODE provide evaluation facilities. WebODE includes a module that performs ontology evaluation according to the OntoClean method [23,35]. OILEd and OntoSaurus are the only ones performing automatic classifications, as they are based on description logic (DL) languages. Finally, none of the tools provide exception-handling mechanisms.

Another important aspect is the *software architecture and tool evolution*, which considers which hardware and software platforms are necessary to use the tool, its architecture (standalone, client/server, n-tier application), extensibility, storage of the ontologies (databases, ASCII files, etc.), failure tolerance, backup management, stability and tool versioning policies. From that perspective, most of the tools are moving towards Java platforms: WebOnto, OILEd, OntoEdit, Protégé 2000 and WebODE. Storage in databases is still a weak point of ontology tools, since just a few of them use databases for storing ontologies: the commercial version of OntoEdit, Protégé 2000 and WebODE. Backup management functionality is just provided by WebODE, and extensibility facilities are just allowed in OntoEdit, Protégé 2000 and WebODE.

Interoperability with other ontology development tools, merging tools, information systems and databases; as well as translations to and from some ontology languages is another important feature in order to integrate ontologies in applications. Most of the new tools export and import to ad-hoc XML and other markup languages. However, there is not a comparative study about the quality of all these translators. Moreover, there are no empirical results about the possibility of exchanging ontologies between different tools and about the amount of knowledge that is lost in the translation processes.

Concerning the *methodology* that the tool gives support to, WebODE gives support to METHONTOLOGY, and OntoEdit gives support to On-To-Knowledge. However, none of the tools analyzed includes: project management facilities, (semi)automatic knowledge acquisition facilities, maintenance and they only provide a little support for ontology verification.

Related to the *cooperative and collaborative construction of ontologies*, WebOnto has the most advanced features. In general, more features are required in existing tools to ensure a successful collaborative building of ontologies. Finally, *Usability aspects* related to help system, edition and visualization, etc., should be improved in most of the tools.

Table 2
A comparison of ontology development tools

	DUET	OILEd	Onto Edit Professional	Ontolingua	OntoSaurus	Protégé 2000	WebODE	WebOnto
General issues								
Developers	AT & T	University of Manchester	Ontoprise	KSL (Stanford University)	ISI (USC)	SMI (Stanford University)	UPM	KMI (Open University)
Current release and date	0.3 (Jul2002)	3.4 (Apr2002)	3.0 (Aug2002)	1.0.649 (Nov2001)	1.9. (Mar2002)	1.8 (Jul2002)	2.0 (Mar2002)	2.3 (May2001)
Pricing policy	Freeware	Freeware	Freeware and licenses	Free Web access	Open source Evaluation version	Open source	Free Web access Licenses	Free Web access
Software architecture								
Sw architecture	Plugin	Standalone	Standalone and client-server	Client/server	Client/server	Standalone	3-tier	Client/server
Extensibility	No	No	Plugins	None	None	Plugins	Plugins	No
Ontology storage	No	File	File DBMS (v3.0)	Files	Files	FileDBMS (JDBC)	DBMS (JDBC)	File
Backup management	No	No	No	No	No	No	Yes	Yes
Interoperability translations to/from languages								
Imports from languages	DAML + OIL	RDF(S), OIL, DAML + OIL	XML RDF(S) FLogic DAML + OIL	Ontolingua IDL KIF	LOOM IDL ONTO KIF C++	XML, RDF(S), XML Schema	XML, RDF(S), CARIN	OCML
Exports to languages	DAML + OIL	OIL RDF(S) DAML + OIL SHIQ Dotty HTML	XML RDF(S) FLogic DAML + OIL SQL-3	KIF-3.0 CLIPS CML ATP CML rule engine EpiKit IDL KSL rule engine LOOM OKBC syntax	LOOM IDL ONTO KIF C++	XML, RDF(S), XML Schema, FLogic, CLIPS, Java HTML	XML, RDF(S) OIL DAML + OIL CARIN FLogic Prolog Jess Java	OCML Ontolingua GXL RDF(S) OIL

Knowledge representation and methodological support								
KR paradigm-of knowledge model	Object oriented classes	DL (DAML + OIL)	Frames + FOL	Frames + FOL (Ontolingua)	DL (LOOM)	Frames + FOL + Meta-classes	Frames + FOL	Frames + FOL
Axiom language	No	Yes (DAML + OIL)	Yes (FLogic)	Yes (KIF)	Yes (LOOM)	Yes (PAL)	Yes (WAB)	Yes (OCML)
Methodological support	Yes (Rational Rose)	No	Yes (Onto-Knowledge)	No	No	No	Yes (METHONTOLOGY)	No
Inference services								
Built-in inference engine	No	Yes (FaCT)	Yes (Onto-Broker)	No	Yes	Yes (PAL)	Yes (Prolog)	Yes
Other attached inference engine	No	No	No	ATP	Yes	JessFaCT FLogic	Jess	No
Constraint/consistency checking	No	Yes	Yes	No	Yes	Yes	Yes	Yes
Automatic classifications	No	Yes	No	No	Yes	No	No	No
Exception handling	No	No	No	No	No	No	No	No
Usability								
Graphical taxonomy	Yes	No	No	Yes	No	Yes	Yes	Yes
Graphical prunes (views)	Yes	No	No	No	No	Yes	Yes	Yes
Zooms	No	No	No	No	No	Yes	No	No
Collaborative working	No	No	Yes	Yes	Yes	No	Yes	Yes
Ontology libraries	No	Yes	Yes	Yes	No	Yes	No	Yes

5. Ontology languages

At the beginning of the 1990s, a set of AI-based ontology implementation languages was created. Basically, the KR paradigm underlying such ontology languages was based on first order logic (i.e. KIF), on frames combined with first order logic (i.e. Ontolingua, OCML and FLogic) or on DL (i.e. Loom).

KIF [24] is a language based on first order logic created in 1992 as an interchange format for diverse KR systems. *Ontolingua* [18,32], which builds on KIF, was developed in 1992 by the KSL at Stanford University. It combines the KR paradigms of frames and first order predicate calculus (KIF). It is the most expressive of all the languages that have been used for representing ontologies, allowing the representation of concepts, taxonomies of concepts, n-ary relations, functions, axioms, instances and procedures. Its high expressiveness led to difficulties in building reasoning mechanisms for it. Hence, no reasoning support is provided with the language.

Loom [54] was developed simultaneously with Ontolingua at the Information Science Institute (ISI) at the University of South California. Initially, it was not meant for implementing ontologies, but for general KBs. Loom is based on DLs and production rules, and provides automatic classifications of concepts. The following ontology components can be represented with this language: concepts, concept taxonomies, n-ary relations, functions, axioms and production rules.

OCML [56] was developed later, in 1993, at the KMI at the Open University. It was created as a kind of "operational Ontolingua". In fact, most of the definitions that can be expressed in OCML are similar to the corresponding definitions in Ontolingua, and some additional components can be defined: deductive and production rules, and operational definitions for functions. OCML was built for developing executable ontologies and models in problem solving methods.

FLogic [46] was developed in 1995 at the Karlsruhe University. FLogic (*Frame Logic*) combines frames and first order logic, allowing to represent concepts, concept taxonomies, binary relations, functions, instances, axioms and deductive rules. FLogic is the only of the previous languages that do not have Lisp-like syntax. Its inference engine, Ontobroker [14], can be used for constraint checking and deducting new information.

In Spring 1997, the High Performance Knowledge Base program (HPKB) started. This research program was sponsored by DARPA, and its objective was to solve many of the problems that usually appear when dealing with large KBs (concerning efficiency, content creation, integration of the content available in different systems, etc.). One of the results of this program was the development of the *OKBC* (*Open Knowledge Base Connectivity*) protocol [10]. This protocol allows accessing KBs stored in different knowledge representation systems (KRSs). Of the systems presented before, Ontolingua and LOOM are OKBC compliant.

The boom of the Internet led to the creation of ontology languages that exploited the characteristics of the Web. Such languages are usually called *web-based ontology languages* or *ontology markup languages*. These languages are still in a development phase: they are continuously evolving. These languages and the relationships among them are shown in Fig. 2.

SHOE [53] was built in 1996 as an extension of HTML, in the University of Maryland. It uses tags different from those of the HTML specification, thus it allows the insertion of ontologies in HTML documents. SHOE combines frames and rules. SHOE just allows representing concepts,

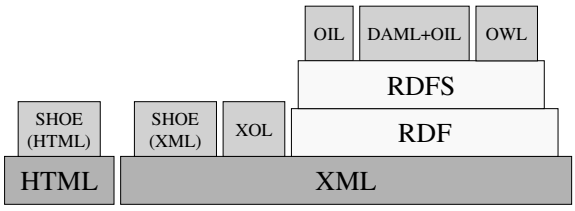

Fig. 2. The stack of ontology markup languages.

their taxonomies, n-ary relations, instances and deduction rules, which are used by its inference engine to obtain new knowledge.

Then, *XML* [7] was created and widely adopted as a standard language for exchanging information on the Web. As a consequence, SHOE syntax was modified to use XML and then, other ontology languages were built on the XML syntax.

XOL [43] was developed by the AI center of SRI international, in 1999, as a XMLization of a small subset of primitives from the OKBC protocol, called OKBC-Lite. It is a very restricted language where only concepts, concept taxonomies and binary relations can be specified. No inference mechanisms are attached to it, as it was mainly designed for the exchange of ontologies in the biomedical domain.

RDF [50] was developed by the W3C (the World Wide Web Consortium) as a semantic-network based language to describe Web resources. *RDF Schema* [8] was built by the W3C as an extension to RDF with frame-based primitives. The combination of both RDF and RDF Schema is normally known as *RDF*(*S*). RDF(S) is not very expressive, just allowing the representation of concepts, concept taxonomies and binary relations. Some inference engines have been created for this language, mainly for constraint checking.

These languages have established the foundations of the Semantic Web. [6] In this context, three more languages have been developed as extensions to RDF(S): OIL, DAML + OIL and OWL.

OIL [38] was developed in the framework of the European IST project On-To-Knowledge. It adds frame-based KR primitives to RDF(S), and its formal semantics is based on DLs. The FaCT classifier is used to perform automatic classifications of concepts.

DAML − ONT specification was released some time later in the context of the DARPA initiative DAML (DARPA Agent Markup Language). On December 2000, it was upgraded to *DAML + OIL* [40], which was created by a joint committee from the US and the EU in the context of the DARPA project DAML. DAML + OIL also adds DL-based KR primitives to RDF(S). Both OIL and DAML + OIL allow representing concepts, taxonomies, binary relations, functions and instances. Many efforts are being put to provide reasoning mechanisms for DAML + OIL.

Finally, in 2001, the W3C formed a working group called Web-Ontology (WebOnt) Working Group. [7] The aim of this group was to make a new ontology markup language for the Semantic Web, called *OWL* (Web Ontology Language). They have already defined a list of main use cases

[6] www.sciam.com/2001/0501issue/0501berners-lee.html.
[7] http://www.w3.org/2001/sw/WebOnt/.

for the Semantic Web, have taken DAML + OIL features as the main input for developing OWL and have proposed the first specification of this language [13]. OWL is divided in two layers: OWLlite and OWL.

Below we present some conclusions of our work with the languages presented above. We have built these conclusions by comparing all these languages with respect to the same evaluation framework [11]. The symbol '+' means that the feature is supported by the language, the symbol '−' means that the feature is not supported by the language, and the symbol '±' means that the feature is not directly supported by the language but it can be represented using a workaround. The results of this comparison are summarized in Table 3. In this table, we do not present the results for OWL, since the specification is only a working draft and could evolve very fast. In the current specification, the values are equivalent to those presented for DAML + OIL.

Concepts, organized in taxonomies, *binary relations* and *instances* are the only components that can be represented in all of the presented languages. However, some differences exist in the primitives available in each language for representing concept taxonomies. In this sense, Ontolingua, LOOM, OCML, OIL, DAML + OIL and OWL are more expressive, since they allow creating exhaustive and disjoint subclass partitions of a concept.

In Ontolingua and SHOE, arbitrary *n-ary relations* can be created. In the rest of languages, these relations must be represented by their decomposition into binary relations.

Functions can be defined easily in Ontolingua, LOOM, OCML, OIL, DAML + OIL and OWL. In FLogic they can be created by defining a relation and an additional axiom that restricts the number of values that it can have.

Formal axioms are the most powerful means of representing knowledge in ontologies, and they are usually used to represent those pieces of knowledge that cannot be represented with other primitives of the language. Formal axioms can be defined in Ontolingua, LOOM, OCML and FLogic.

Finally, *rules* can only be defined in LOOM and OCML, and *procedures* can only be defined in Ontolingua (although they cannot be executed), LOOM and OCML.

Concerning the *inference* mechanisms attached to each language, they are diverse. Except for the OIL inference engine (FaCT), inference engines are used to deduce new knowledge from the ontology or check inconsistencies with its formal axioms. In LOOM and OIL, the inference engine also performs automatic concept classifications.

In summary, KR paradigms underlying all the languages are diverse: frames, DLs, first (and second) order predicate calculus, conceptual graphs, semantic networks, production rules, deductive rules, etc. In many cases, they are based on combinations of several formalisms. Additionally, there is a tight interdependence between expressiveness and reasoning in all the languages, in the sense that the expressive power of a language must be sometimes limited to ensure a good reasoning service, as shown in [11].

The main lesson to learn from this study is that if we need to implement an ontology, we should decide first what our application needs in terms of expressiveness and inference services, because not all of the existing languages allow representing the same components and reason in the same way. The representation and reasoning with basic information, such as concepts, taxonomies and binary relations, is not usually enough if we want to create a heavyweight ontology and make complex reasonings with it, and existing translations between languages are not good enough yet

Table 3
A comparison of ontology languages

	Onto-lingua	OCML	LOOM	FLogic	XOL	SHOE	RDF(S)	OIL	DAML
Concepts									
General issues									
Metaclasses	+	+	+	+	+	−	+	−	−
Partitions	+	±	+	±	−	−	−	+	+
Documentation	+	+	+	±	+	+	+	+	+
Attributes									
Template (instance attributes)	+	+	+	+	+	+	+	+	+
Own (class attributes)	+	+	+	+	+	−	−	+	+
Local scope	+	+	+	+	+	+	+	+	+
Global scope	±	±	+	−	+	−	+	+	+
Facets									
Default slot value	−	+	+	+	+	−	−	−	−
Type constraint	+	+	+	+	+	+	+	+	+
Cardinality constraints	+	+	+	±	+	−	−	+	+
Slot documentation	+	+	+	−	+	+	+	+	+
Taxonomies									
Subclass of	+	+	+	+	+	+	+	+	+
Exhaustive subclass partitions	+	±	+	±	−	−	−	+	+
Disjoint decompositions	+	±	+	±	−	−	−	+	+
Not subclass of	±	−	±	−	−	−	−	+	+
Relations and functions									
n-ary relations/functions	+	+	+	±	±	+	±	±	±
Type constraints	+	+	+	+	+	+	+	+	+
Integrity constraints	+	+	+	+	−	−	−	−	−
Operational definitions	−	+	+	+	−	−	−	−	−
Axioms									
First order logic	+	+	+	+	−	±	−	±	±
Second order logic	+	−	−	−	−	−	−	−	−
Named axioms	+	+	−	−	−	−	−	−	−
Embedded axioms	+	+	+	−	−	−	−	−	−
Instances									
Instances of concepts	+	+	+	+	+	+	+	+	+
Facts	+	+	+	+	+	+	+	+	+
Claims	−	−	−	−	−	+	±	±	±

to ensure that information is not lost in the process. Hence, making a good decision of using a specific language for representing ontologies is crucial for developing an ontology-based application.

6. Conclusions

Fig. 3 presents the relationships between the main methods and methodologies, tools and languages that have been presented in this paper. From this study, we can extract the following conclusions:

- There is no correspondence between ontology building methodologies and tools, except for METHONTOLOGY and WebODE, and On-To-Knowledge and OntoEdit. Since there is no technological support for most of the existing methodologies, they cannot be easily applied in the ontology construction task. In fact, most of the tools just focus on few activities of the ontology lifecycle: design and implementation.
- There are many "similar" ontology building tools available. However, they are not usually able to interoperate, which provokes important problems when integrating ontologies in the ontology library of a different tool, when merging ontologies available in different ontology tools or languages, etc.
- Nowadays, it is not usually necessary to implement ontologies manually, as most of the available ontology tools are able to generate ontologies in many different ontology languages.
- Ontology markup languages are still in development phases, and they are continuously evolving, which makes it difficult to have up-to-date technology for managing them.

If we focus on ontology tools, the main problem to be solved in this area is the lack of integrated environments for ontology development (except for some environments such as OntoEdit, Protégé 2000 or WebODE). Tools are usually created as isolated modules that solve one type of problems, but are not fully integrated with other activities of the ontology lifecycle. Consequently, future work should be driven towards the creation of a *common workbench for ontology development* (as shown in Fig. 4) that facilitates:

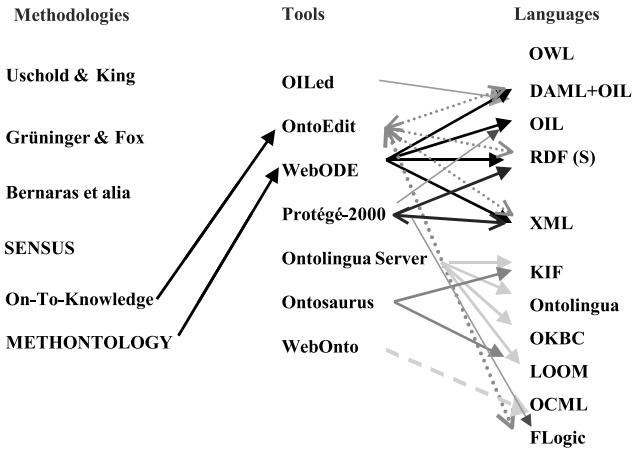

Fig. 3. Ontology methodologies, tools and languages.

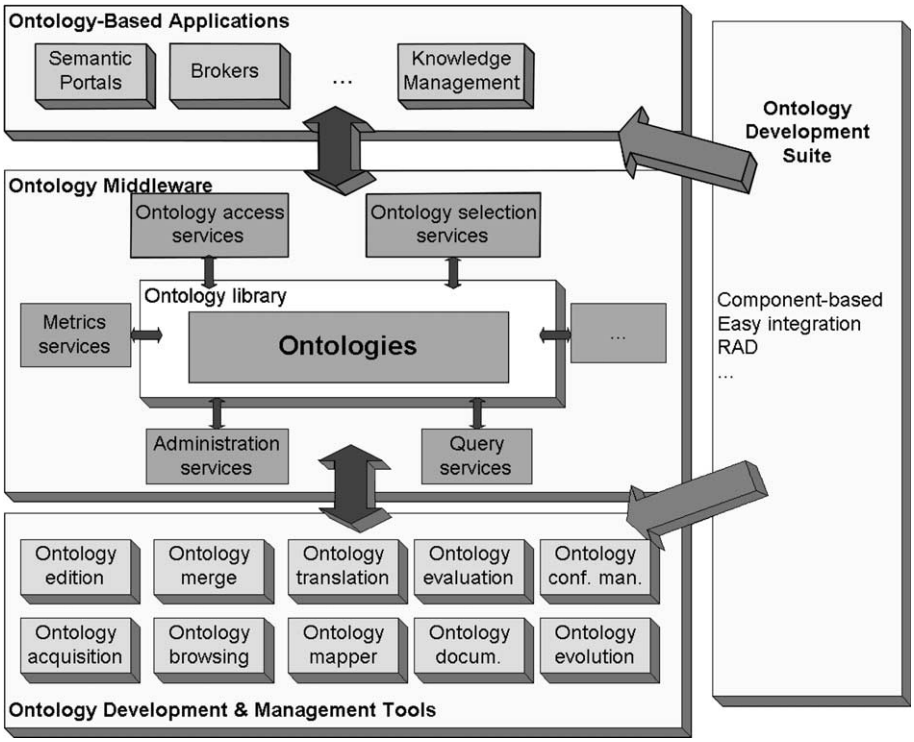

Fig. 4. A proposed workbench for ontology development and use.

- Ontology development during the whole ontology life cycle, including: knowledge acquisition, edition, browsing, integration, merging, ontological mappings, reengineering, evaluation, translation to different languages and formats, interchange of content with other tools, etc.
- Ontology management: configuration management and evolution of isolated ontologies as well as of ontology libraries.
- Ontology support: scheduling, documentation, advanced techniques for visualising the ontology content, etc.
- Methodological support for building ontologies.

This ontology development workbench should be also accompanied by a set of *ontology middleware services* that support the use of ontologies in other systems. Some of these services are:

- Software that helps to locate the most appropriate ontology for a given application.
- Formal metrics that compare the semantic similarity and semantic distance between terms of the same or different ontologies.
- Software that allows incremental, consistent and selective upgrades of the ontology which is being used by a given application.
- Query modules to consult the ontology.
- Remote access to the ontology library system.
- Software that facilities the integration of the ontology with legacy systems and databases.

Finally, a wide transfer of this technology into companies, with the subsequent development of a large number of ontology-based applications in the Semantic Web context, will be achieved by the creation of *ontology application development suites*, which will allow the rapid development and integration of existing and future applications in a component based basis.

URLs

Tools

DUET: http://codip.grci.com/Tools/Tools.html.
OILEd: http://img.cs.man.ac.uk/oil/.
Ontolingua Server: http://ontolingua.stanford.edu/.
OntoSaurus: http://www.isi.edu/isd/ontosaurus.html.
OntoEdit: http://ontoserver.aifb.uni-karlsruhe.de/ontoedit/.
Protégé 2000: http://protege.stanford.edu/.
WebODE: http://delicias.dia.fi.upm.es/webODE/.
WebOnto: http://webonto.open.ac.uk/.

Projects

OntoWeb SIG on Enterprise-Standard Ontology Environments: http://delicias.dia.fi.upm.es//ontoweb-fac/sig-tools/.
OntoWeb SIG on Ontology Language Standards: http://www.cs.man.ac.uk/~horrocks/OntoWeb/SIG/.
OntoWeb Homepage: http://www.ontoweb.org/.

Acknowledgements

This work has been partially supported by the IST thematic network OntoWeb (IST-2000-29243), the IST project Esperonto (IST-2001-34373), the CICYT project "ContentWeb: Plataforma Tecnológica para la Web Semántica" (TIC-2001-2745) and a FPU (*Formación de Personal Universitario*) grant from UPM.

References

[1] J.C. Arpírez, O. Corcho, M. Fernández-López, A. Gómez-Pérez, WebODE: a scalable ontological engineering workbench, in: First International Conference on Knowledge Capture (KCAP'01), ACM Press, Victoria, 2001, pp. 6–13.
[2] N. Aussenac-Gilles, B. Biébow, S. Szulman, Revisiting ontology design: a methodology based on corpus analysis, in: 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00), Lecture Notes in Artificial Intelligence, vol. 1937, Springer, Berlin, 2000, pp. 172–188.
[3] S. Bechhofer, I. Horrocks, C. Goble, R. Stevens, OilEd: a reason-able ontology editor for the Semantic Web, in: Joint German/Austrian conference on Artificial Intelligence (KI'01), Lecture Notes in Artificial Intelligence, vol. 2174, Springer, Berlin, 2001, pp. 396–408.
[4] A. Bernaras, I. Laresgoiti, J. Corera, Building and reusing ontologies for electrical network applications, in: Proc. European Conference on Artificial Intelligence (ECAI'96), Budapest, Hungary, 1996, pp. 298–302.

[5] M. Blázquez, M. Fernández-López, J.M. García-Pinar, A. Gómez-Pérez, Building ontologies at the knowledge level using the ontology design environment, in: B.R. Gaines, M.A. Musen (Eds.), 11th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'98) Banff, 1998.
[6] W.N. Borst, Construction of Engineering Ontologies, PhD Thesis, University of Tweenty, Enschede, NL—Centre for Telematica and Information Technology, 1997.
[7] T. Bray, J. Paoli, CM. Sperberg-McQueen, E. Maler, Extensible Markup Language (XML) 1.0, second ed., W3C Recommendation, 2000. Available from <http://www.w3.org/TR/REC-xml>.
[8] D. Brickley, R.V. Guha, RDF Vocabulary Description Language 1.0: RDF Schema, W3C Working Draft, 2002. Available from <http://www.w3.org/TR/PR-rdf-schema>.
[9] H. Chalupsky, OntoMorph: a translation system for symbolic knowledge, in: Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR2000), Morgan Kaufmann, San Francisco, 2000, pp. 471–482.
[10] V.K. Chaudhri, A. Farquhar, R. Fikes, P.D. Karp, J.P. Rice, Open Knowledge Base Connectivity 2.0.3, Technical Report, 1998. Available from <http://www.ai.sri.com/~okbc/okbc-2-0-3.pdf>.
[11] O. Corcho, A. Gómez-Pérez, A roadmap to ontology specification languages, in: 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00), Lecture Notes in Artificial Intelligence, vol. 1937, Springer, Berlin, 2000, pp. 80–96.
[12] O. Corcho, M. Fernández-López, A. Gómez-Pérez, O. Vicente, WebODE: an integrated workbench for ontology representation, reasoning and exchange, in: A. Gómez-Pérez, V.R. Benjamins (Eds.), 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02), Lecture Notes in Artificial Intelligence, vol. 2473, Springer, Berlin, 2002, pp. 138–153.
[13] M. Dean, D. Connolly, F. van Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, L.A. Stein, OWL Web Ontology Language 1.0 Reference, W3C Working Draft, 2002. Available from <http://www.w3.org/TR/owl-ref/>.
[14] S. Decker, M. Erdmann, D. Fensel, R. Studer, Ontobroker: Ontology based access to distributed and semi-structured information, in: Semantic Issues in Multimedia Systems (DS8), Kluwer Academic Publisher, Boston, 1999, pp. 351–369.
[15] J. Domingue, Tadzebao and Webonto: Discussing, Browsing and Editing Ontologies on the Web, in: Proc. 11th Knowledge Acquisition Workshop (KAW98), Banff, 1998.
[16] A. Duineveld, R. Studer, M. Weiden, B.Kenepa, R. Benjamis, WonderTools? A comparative study of ontological engineering tools, in: Proc. 12th Knowledge Acquisition Workshop (KAW99), Banff, 1999.
[17] J. Euzenat, Corporative memory through cooperative creation of knowledge bases and hyper-documents, in: Proc. 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW96), Banff, 1996.
[18] A. Farquhar, R. Fikes, J. Rice, The Ontolingua Server: A Tool for Collaborative Ontology Construction, in: Proc. 10th Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW96) Banff, 1996, pp. 44.1–44.19.
[19] M. Fernández-López, A. Gómez-Pérez, M.D. Rojas-Amaya, Ontologies' crossed life cycles, in: Proc. 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00), Lecture Notes in Artificial Intelligence, vol. 1937, Springer, Berlin, 2000, pp. 65–79.
[20] M. Fernández-López, A. Gómez-Pérez, A. Pazos-Sierra, J. Pazos-Sierra, Building a chemical ontology using METHONTOLOGY and the ontology design environment, IEEE Intelligent Systems & their applications 4 (1) (1999) 37–46.
[21] M. Fernández-López, Overview of Methodologies for Building Ontologies, in: IJCAI99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends, Stockholm, 1999.
[22] M. Fernández-López, A. Gómez-Pérez, N. Juristo, METHONTOLOGY: From Ontological Art Towards Ontological Engineering, AAAI Symposium on Ontological Engineering, Stanford, 1997.
[23] A. Gangemi, N. Guarino, A. Oltramari, Conceptual analysis of lexical taxonomies: the case of wordnet top-level, in: International Conference on Formal Ontology in Information Systems (FOIS'01), ACM Press, Maine, 2001, pp. 3–15.
[24] M. Genesereth, R. Fikes, Knowledge interchange format, Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.

[25] A. Gómez-Pérez, A framework to verify knowledge sharing technology, Expert Systems with Application 11 (4) (1996) 519529.
[26] A. Gómez-Pérez, Knowledge sharing and reuse, in: J. Liebowitz (Ed.), Handbook of Expert Systems, CRC, New York, 1998, Chapter 10.
[27] A. Gómez-Pérez, Evaluation of ontologies, International Journal of Intelligent Systems 16 (3) (2001)10:1–10: 36.
[28] A. Gómez-Pérez, M. Fernández-López, A. de Vicente, Towards a Method to Conceptualize Domain Ontologies, in: ECAI96 Workshop on Ontological Engineering, Budapest, 1996, pp. 41–51.
[29] A. Gómez-Pérez, M.D. Rojas, Ontological reengineering and reuse, in: D. Fensel, R. Studer (Eds.), 11th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW'99), Lecture Notes in Artificial Intelligence, vol. 1621, Springer, Berlin, 1999, pp. 139–156.
[30] A. Gómez-Pérez, A Survey on Ontology Tools, OntoWeb deliverable 1.3, 2001.
[31] T.R. Gruber, A translation approach to portable ontology specification, Knowledge Acquisition 5 (1993) 199–220.
[32] T.R. Gruber. ONTOLINGUA: A Mechanism to Support Portable Ontologies, technical report, Knowledge Systems Laboratory, Stanford University, 1992.
[33] M. Grüninger, M.S. Fox, Methodology for the design and evaluation of ontologies, in: Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, 1995.
[34] N. Guarino, M. Carrara, P. Giaretta, Ontologies and knowledge bases: towards a terminological clarification, in: N. Mars (Ed.), Towards Very Large Knowledge Bases, Knowledge Building and Knowledge Sharing, IOS Press, Amsterdam, 1995, pp. 25–32.
[35] N. Guarino, C. Welty, Ontological analysis of taxonomic relationships, in: 19th International Conference on Conceptual Modeling (ER'00), Lecture Notes in Computer Science, vol. 1920, Springer, Berlin, 2000, pp. 210–224.
[36] N. Guarino, C. Welty, A formal ontology of properties, in: 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00), Lecture Notes in Artificial Intelligence, vol. 1937, Springer, Berlin, 2000, pp. 97–112.
[37] M. Hermenegildo, F. Bueno, D. Cabeza, M. Carro, M. García, P. López, G. Puebla, The Ciao Logic Programming Environment, in: International Conference on Computational Logic (CL2000) 2000.
[38] I. Horrocks, D. Fensel, F. Harmelen, S. Decker, M. Erdmann, M. Klein, OIL in a Nutshell, in: ECAI'00 Workshop on Application of Ontologies and PSMs, Berlin, 2000.
[39] I. Horrocks, U. Sattler, S. Tobies, Practical reasoning for expressive description logics, in: 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99), Lecture Notes in Artificial Intelligence, Springer, Berlin, 1999, pp. 161–180.
[40] I. Horrocks, F. van Harmelen, Reference Description of the DAML + OIL (March 2001) Ontology Markup Language, Technical report, 2001. Available from <http://www.daml.org/2001/03/reference.html>.
[41] Y. Kalfoglou, D. Robertson, Managing Ontological Constraints, in: Proc. IJCAI99 Workshop on Ontologies and Problem-Solving Methods: Lessons Learned and Future Trends, Stockholm, 1999.
[42] Y. Kalfoglou, D. Robertson, Use of formal ontologies to support error checking in specifications, in: D. Fensel, R. Studer (Eds.), 11th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW'99), Lecture Notes in Artificial Intelligence, vol. 1621, Springer, Berlin, 1999, pp. 207–224.
[43] R. Karp, V. Chaudhri, J. Thomere, XOL: An XML-Based Ontology Exchange Language, technical report, 1999. Available from <http://www.ai.sri.com/~pkarp/xol/xol.html>.
[44] E.F. Kendall, M.E. Dutra, D.L. McGuinness, Towards a commercial ontology development environment, in: First International Semantic Web Conference (ISWC'02), Lecture Notes in Computer Science, vol. 2342, Springer, Berlin, 2002.
[45] JU. Kietz, A. Maedche, R. Volz, A method for semi-automatic ontology acquisition from a corporate intranet, in: EKAW'00 Workshop on Ontologies and Texts, CEUR Workshop Proceedings, Juan-Les-Pins, vol. 51 (2000) 4.14.14.

[46] M. Kifer, G. Lausen, J. Wu, Logical foundations of object-oriented and frame-based languages, Journal of the ACM 42 (4) (1995) 741–843.
[47] M. Klein, D. Fensel, Ontology versioning on the Semantic Web, in: First International Semantic Web Workshop (SWWS01), Stanford, 2001.
[48] M. Klein, D. Fensel, A. Kiryakov, D. Ognyanov, Ontology versioning and change detection on the Web, in: A. Gómez-Pérez, V.R. Benjamins (Eds.), 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Lecture Notes in Artificial Intelligence, vol. 2473, Springer, Berlin, 2002.
[49] P. Kogut, S. Cranefield, L. Hart, M. Dutra, K. Baclawski, M. Kokar, J. Smith, UML for ontology development, The Knowledge Engineering Review 17 (1) (2002) 61–64.
[50] O. Lassila, R. Swick, Resource description framework (RDF) model and syntax specification, W3C Recommendation (1999), http://www.w3.org/TR/REC-rdf-syntax/.
[51] O. Lassila, D. McGuinness, The Role of Frame-Based Representation on the Semantic Web, Technical Report KSL-01-02, Knowledge Systems Laboratory, Stanford University, 2001.
[52] D.B. Lenat, R.V. Guha, Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project, Addison-Wesley, Boston, 1990.
[53] S. Luke, J. Heflin, SHOE 1.01. Proposed Specification, SHOE Project technical report, University of Maryland, 2000. Available from <http://www.cs.umd.edu/projects/plus/SHOE/spec1.01.htm>.
[54] R. MacGregor, Inside the LOOM clasifier, SIGART bulletin 2 (3) (1991) 70–76.
[55] D.L. McGuinness, R. Fikes, J. Rice, S. Wilder, The Chimaera Ontology Environment, in: 17th National Conference on Artificial Intelligence (AAAI'00), Austin, 2000.
[56] E. Motta, Reusable Components for Knowledge Modelling, IOS Press, Amsterdam, 1999.
[57] R. Neches, R.E. Fikes, T. Finin, T.R. Gruber, T. Senator, W.R. Swartout, Enabling technology for knowledge sharing, AI Magazine 12 (3) (1991) 36–56.
[58] N.F. Noy, M. Klein, Ontology Evolution: Not the Same as Schema Evolution, Technical Report SMI-2002-0926, Stanford, 2002.
[59] N.F. Noy, M.A. Musen, PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment, in: 17th National Conference on Artificial Intelligence (AAAI'00), Austin, 2000.
[60] N.F. Noy, R.W. Fergerson, M.A. Musen, The knowledge model of protege-2000: combining interoperability and flexibility, in: 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00), Lecture Notes in Artificial Intelligence, vol. 1937, Springer, Berlin, 2000, pp. 17–32.
[61] Ath. Schreiber, B. Wielinga, W. Jansweijer, The KACTUS view on the 'O' word. Technical Report, ESPRIT Project 8145 KACTUS, University of Amsterdam, The Netherlands, 1995.
[62] S. Staab, H.P. Schnurr, R. Studer, Y. Sure, Knowledge processes and ontologies, IEEE Intelligent Systems 16 (1) (2001) 26–34.
[63] L. Stojanovic, B. Motik, Ontology evolution with ontology, in: EKAW02 Workshop on Evaluation of Ontology-based Tools (EON2002), CEUR Workshop Proceedings, Sigüenza, vol. 62, 2002, pp. 53–62.
[64] R. Studer, V.R. Benjamins, D. Fensel, Knowledge engineering: principles and methods, Data and Knowledge Engineering 25 (1998) 161–197.
[65] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, D. Wenke, OntoEdit: collaborative ontology engineering for the semantic web, in: First International Semantic Web Conference (ISWC'02), Lecture Notes in Computer Science, vol. 2342, Springer, Berlin, 2002, pp. 221–235.
[66] B. Swartout, P. Ramesh, K. Knight, T. Russ, Toward Distributed Use of Large-Scale Ontologies, AAAI Symposium on Ontological Engineering, Stanford, 1997.
[67] M. Uschold, Building Ontologies: Towards A Unified Methodology, Expert Systems, Cambridge, 1996.
[68] M. Uschold, M. Grüninger, Ontologies: Principles methods and applications, The Knowledge Engineering Review 11 (2) (1996) 93–155.
[69] M. Uschold, R. Jasper, A Framework for Understanding and Classifying Ontology Applications, in: Proc. IJCAI99 Workshop on Ontologies and Problem-Solving Methods, Stockholm, 1999.
[70] M. Uschold, M. King, Towards a Methodology for Building Ontologies, in: IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, 1995.

MSc. Oscar Corcho received his BA in Computer Science (2000) and his M.Sc. on Software Engineering (2001) from the Computer Science School at Universidad Politécnica de Madrid, Spain. He received the third Spanish award in Computer Science by Spanish Government (2001). He belongs to the Ontology Group of the Artificial Intelligence Laboratory at the Computer Science School at UPM. His research activities include ontology languages, translation problem and the Semantic Web. He visited the Stanford Medical Informatics department (SMI) at Stanford University for two months in 2002, and was an invited consultant in the KMI of the Open University (England) for two months in 1999. He participates at the OntoWeb thematic network and the IST Esperonto and MKBEEM projects. His papers have been published in important conferences, workshops and journals for the ontology community. He also acts as reviewer in some workshops and conferences, and has organized the demo/industrial track session at the EKAW 2002 conference.

Dr. Mariano Fernández-López is Assistant Professor at the Computer Science School at UPM. He received his BA in Computer Science (1996), M.Sc. on Knowledge Engineering (2000), M.Sc. on Software Engineering (2000), a Ph.D. degree cum laude in Computer Sciences (2001) by UPM. He was Teaching Assistant at the Centro de Estudios Universitarios (CEU) (1998–1999), and Invited Professor in the Universidad Pontificia de Salamanca. He belongs to the Ontology Group since 1995. His current research activities include, among others: Ontological Engineering, and Electronic Commerce. He has published in different national and international forums and journals. He is currently involved in several national and international research projects (for example, the thematic network OntoWeb and the IST projects Esperonto and MKBEEM). He is the tutorial and workshop organiser in the EKAW-2002 conference, and he is member of the committee of EKAW-2002 and JIISIC-01. He lectures a Ph.D. course on ontologies at the AI Department at UPM.

Dr. Asunción Gómez-Pérez is an Associate Professor at the Computer Science School at Universidad Politécnica de Madrid, Spain. She is BA in Computer Science (1990), M.Sc. on Knowledge Engineering (1991), Ph.D. in Computer Sciences (1993) by Universidad Politécnica de Madrid (UPM), Spain. She is also M.Sc. on Business Administration (1994) by Universidad Pontificia de Comillas, Spain. She was visiting (1994–1995) the KSL at Stanford University. She was also the Executive Director (1995–1998) of the AI Lab at the School. Currently, she is advisor for research at the same Lab and she is the director of the Ontology group since 1995. She is participating as a main node and member of the Executive Program Board Committee at the Ontoweb thematic network and also at the IST Esperonto and MKBEEM projects. Her current research activities include, among others: Theoretical ontological foundations, Methodologies for building and merging ontologies, Ontological Reengineering, Uses of ontologies in applications related with e-commerce and Knowledge Management, Semantic Web, etc. She has published more than 60 papers on the above issues. She has leaded several national and international projects related with ontologies funded by various institutions and/or companies related. She has chaired the EKAW-2002 conference. She has been co-organizer of the workshops and conferences on ontologies at IJCAI-01, ECAI-00, IJCAI-99, ECAI-98, SSS-97 and ECAI-96. She has taught tutorials on Ontological Engineering at ECAI-98, SEKE-97 and CAEPIA-97. She acts as reviewer in many conferences and journals.