Chapter III
Ontological Engineering:
What Are Ontologies and How Can We Build Them?

Oscar Corcho
University of Manchester, UK

Mariano Fernández-López
Universidad San Pablo CEU and Universidad Politécnica de Madrid, Spain

Asunción Gómez-Pérez
Universidad Politécnica de Madrid, Spain

ABSTRACT

Ontologies are formal, explicit specifications of shared conceptualizations. There is much literature on what they are, how they can be engineered and where they can be used inside applications. All these literature can be grouped under the term "Ontological Engineering," which is defined as the set of activities that concern the ontology development process, the ontology lifecycle, the principles, methods and methodologies for building ontologies, and the tool suites and languages that support them. In this chapter we provide an overview of Ontological Engineering, describing the current trends, issues and problems.

INTRODUCTION

The origin of ontologies in computer science can be referred back to 1991, in the context of the DARPA Knowledge Sharing Effort (Neches, Fikes, Finin, Gruber, Senator, & Swartout, 1991). The aim of this project was to devise new ways of constructing knowledge-based systems, so that the knowledge bases upon which they are based did not have to be constructed from scratch, but by assembling reusable components. This reuse applies both to static knowledge, which is modeled by means of ontologies, and dynamic problem-solving knowledge, which is modeled by means of problem solving methods.

Since then, considerable progress has been made in this area. Ontologies are now considered as a commodity that can be used for the development of a large number of applications in different fields, such as knowledge management, natural language processing, e-commerce, intelligent integration information, information retrieval, database design and integration, bio-informatics, education, and so forth.

The emergence of the Semantic Web (Berners-Lee, 1999) has caused a growing need for knowledge reuse, and has strenghtened its potential at the same time. Therefore, ontologies and problem-solving methods (which in some cases are considered as the precursors of Semantic Web Services) are playing an important role in this context.

As described in the chapter title, we will present the *what* and *how* of ontologies, describing the activities that should be carried out during the ontology development process, the principles to be followed in ontology design, and the methods, methodologies, software tools and languages that give support to each one of these activities. The second section defines the word "ontology" and explains which are the main components that can be used to model ontologies. The third section focuses on methods and methodologies for the development of ontologies, either used for the whole ontology development process or only for specific activities. The fourth section focuses on ontology tools, which normally give support to the previous methodological approaches. The fifth section describes ontology languages that can be used to implement ontologies. All these sections are structured in a similar way: first we give a brief overview of their evolution, then we describe the current trends, and finally we pay attention to the open issues and practical aspects. Finally, conclusions and future lines of work are presented in the last section.

WHAT IS AN ONTOLOGY AND WHICH ARE ITS COMPONENTS?

There are two different views about the use of the term "ontology," considering whether the person who uses that term is interested in its philosophical roots or in its application to Computer Science.

For philosophers, the term *Ontology* (normally typed with uppercase) refers to the "the essence of things through the changes." Greek philosophers, from Parmenides of Elea to Aristotle, were interested in these aspects. In the 18th century, Kant worked also on these ideas. More recently, people working in the area of formal ontologies are also interested in these philosophical ideas and its application in the context of Computer Science.

On the other side, ontology engineers in the context of computer science are more interested in how ontologies (typed with lowercase) can be used to represent reusable and sharable pieces of domain knowledge and how they can be used in applications. In this context, ontologies are reusable and sharable artifacts that have to be developed in a machine interpretable language (Gruber, 1993; Studer, Benjamins, & Fensel, 1998). This point of view is clearly addressed in the definition given by Studer and colleagues (1998): *An ontology is a formal, explicit specification of a shared conceptualization.* We consider that this definition is one of the most complete ones from those available in the literature.

Once we have analysed these different definitions of the term "ontology," we will focus on the second use of this term, that is, on what is normally known as Ontological Engineering (Gómez-Pérez, Fernández-López, & Corcho, 2003). First we will discuss about the components that are used to create an ontology.

Different knowledge representation formalisms (and corresponding languages) exist for the

fomalisation (and implementation) of ontologies. Each of them provides different components that can be used for these tasks. However, they share the following minimal set of components.[1]

Classes represent concepts, which are taken in a broad sense. For instance, in the traveling domain, concepts are: locations (cities, villages, etc.), lodgings (hotels, camping, etc.) and means of transport (planes, trains, cars, ferries, motorbikes and ships). Classes in the ontology are usually organised in taxonomies through which inheritance mechanisms can be applied. We can represent a taxonomy of entertainment places (theater, cinema, concert, etc.) or travel packages (economy travel, business travel, etc.). In the frame-based KR paradigm, metaclasses can also be defined. Metaclasses are classes whose instances are classes. They usually allow for gradations of meaning, since they establish different layers of classes in the ontology where they are defined.

Relations represent a type of association between concepts of the domain. They are formally defined as any subset of a product of n sets, that is: $R \subset C1 \times C2 \times ... \times Cn$. Ontologies usually contain binary relations. The first argument is known as the domain of the relation, and the second argument is the range. For instance, the binary relation arrivalPlace has the concept Travel as its domain and the concept Location as its range. Relations can be instantiated with knowledge from the domain. For example, to express that the flight AA7462-Feb-08-2002 arrives in Seattle we must write: (arrivalPlace AA7462-Feb-08-2002 Seattle)

Binary relations are sometimes used to express concept attributes (i.e., slots). Attributes are usually distinguished from relations because their range is a datatype, such as *string*, *number*, and so forth, while the range of relations is a concept. The following code defines the attribute flightNumber, which is a *string*. We can also express relations of higher arity, such as "a road connects two different cities."

According to Gruber (1993), *formal axioms* serve to model sentences that are always true. They are normally used to represent knowledge that cannot be formally defined by the other components. In addition, formal axioms are used to verify the consistency of the ontology itself or the consistency of the knowledge stored in a knowledge base. Formal axioms are very useful to infer new knowledge. An axiom in the traveling domain would be that it is not possible to travel from the America to Europe by train.

Instances are used to represent elements or individuals in an ontology. An example of instance of the concept `AA7462` is the flight AA7462 that arrives at Seattle on February 8, 2006 and costs 300 (US Dollars, Euros, or any other currency).

Besides formalisms and languages specifically designed for representing knowledge, ontologies can be formalised with other approaches coming from the areas of Software Engineering, such as the Unified Modeling Language (UML) (Rumbaugh, Jacobson, & Booch, 1998) or Entity-Relationship (ER) Diagrams (Chen, 1976).

In this context, the Object Management Group (OMG)[2] is working on a specification to define the meta-models of some of the diagram types and languages used in ontology representation. This specification is known as ontology description model (ODM, 2005), and uses a common formal notation to describe the metamodels. Such metamodels (defined for UML, Entity-Relationship, OWL, RDF(S), etc.) can be considered formalisations of knowledge representation ontologies. All these correspondences are formally described in the ODM document (ODM, 2005).

The purpose of ODM documents is to allow software engineers to model ontologies with familiar notations for them, for example, UML and ER, and to transform their conceptual models into formal ontologies represented in ontology languages.

METHODS AND METHODOLOGIES FOR THE DEVELOPMENT OF ONTOLOGIES

Several proposals for ontology development have been reported in the literature. In 1990, Lenat and Guha published the general steps (Lenat & Guha, 1990) 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 (Uschold & King, 1995) and the TOVE (TOronto Virtual Enterprise) project ontology (Grüninger & Fox, 1995) (both in the domain of enterprise modeling), the first guidelines were proposed and later refined in (Uschold, 1996; Uschold & Grüninger, 1996). At the 12th European Conference for Artificial Intelligence (ECAI'96), Bernaras and colleagues (Bernaras, Laresgoiti, & Corera, 1996) presented a method used to build an ontology in the domain of electrical networks as part of the Esprit KACTUS (Schreiber, Wielinga, & Jansweijer, 1995) project. The methodology methontology (Gómez-Pérez, Fernández-López, & de Vicente, 1996) appeared at the same time and was extended in later papers (Fernández-López, Gómez-Pérez, & Juristo, 1997; Fernández-López, Gómez-Pérez, Pazos, & Pazos, 1999). In 1997, a new method was proposed for building ontologies based on the SENSUS ontology (Swartout, Ramesh, Knight, & Russ, 1997). Some years later, the on-to-knowledge methodology appeared as a result of the project with the same name (Staab, Schnurr, Studer, & Sure, 2001). A comparative and detailed study of these methods and methodologies can be found in (Fernández-López & Gómez-Pérez, 2002a).

All the previous methods and methodologies were proposed for building ontologies. However, many other methods have been proposed for specific tasks of the ontology development process, such as ontology reengineering (Gómez-Pérez & Rojas, 1999), ontology learning (Aussenac-Gilles, Biébow, Szulman, 2000a; Kietz, Maedche, & Volz, 2000), ontology evaluation (Gómez-Pérez, 1994, 1996, 2001, 2004; Guarino, 2004; Guarino & Welty, 2002; Kalfoglou & Robertson, 1999a, 1999b; Welty & Guarino, 2001), ontology evolution (Klein & Fensel, 2001; Klein, Fensel, Kiryakov, & Ognyanov, 2002; Noy & Klein, 2002; Noy & Musen, 2004a, 2004b; Noy, Kunnatur, Klein, & Musen, 2004; Stojanovic, 2004), ontology alignment (Benebentano et al., 2000; Castano, De Antonellis, & De Capitani diVemercati, 2001; Ehring & Staab, 2004; Euzenat, 2004; Madhavan, Bernstein, & Rahm, 2001; Melnik, García-Molina, & Rahm, 2002; Noy & Musen, 2001; Pan, Ding, Yu, & Peng, 2005; Shvaiko, Giunchiglia, & Yatskevich, 2004), and ontology merging (Gangemi, Pisanelli, & Steve, 1999; Steve, Gangemi, & Pisanelli, 1998) (Noy & Musen, 2000; Stumme & Maedche, 2001), among others.

In the following subsections we will describe what we understand by ontology development process and ontology lifecycle. Then we will describe the methods and methodologies used for the whole ontology development process. And finally we will focus on ontology learning, ontology merging, ontology alignment, ontology evolution and versioning, and ontology evaluation.

Ontology Development Process and Lifecycle

The ontology development process and the ontology lifecycle were identified by Fernández-López and colleagues (1997) in the framework of methontology. These proposals were based on the IEEE standard for software development (IEEE, 1996).

The ontology development process refers to the activities that have to be performed when building ontologies. They can be classified in three categories (Figure 1):

Ontology management activities include scheduling, control and quality assurance. The *scheduling* activity identifies the tasks to be performed, their arrangement, and the time and

resources needed for their completion. This activity is essential for ontologies that use ontologies stored in ontology libraries or for ontologies that require a high level of abstraction and generality. The *control* activity guarantees that scheduled tasks are completed in the manner intended to be performed. Finally, the *quality assurance* activity assures that the quality of each and every product output (ontology, software and documentation) is satisfactory.

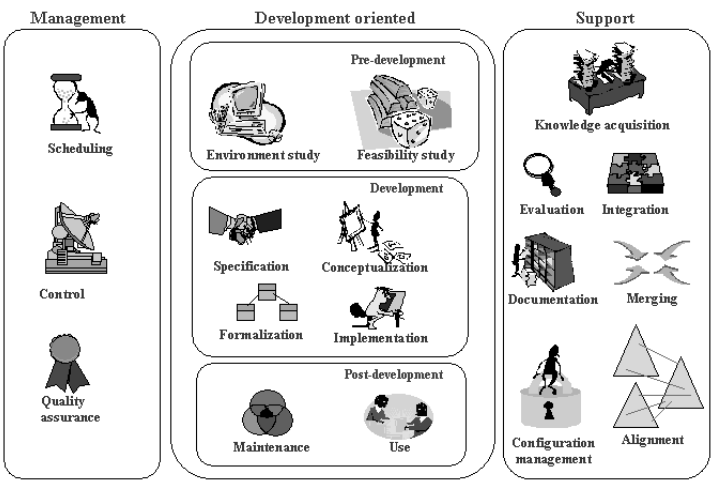
Ontology development oriented activities are grouped, as presented in Figure 1, into predevelopment, development and postdevelopment activities. During the predevelopment, an *environment study* identifies the problem to be solved with the ontology, the applications where the ontology will be integrated, and so forth. Also during the predevelopment, the *feasibility study* answers questions like: is it possible to build the ontology?; is it suitable to build the ontology?; and so forth.

Once in the development, the *specification* activity[3] states why the ontology is being built, what its intended uses are and who the end-users are. The *conceptualisation* activity structures the domain knowledge as meaningful models either from scratch or reusing existing models. In this last case, related activities like pruning branches of the existing taxonomies, extending the coverage of ontologies with the addition of new concepts in the higher levels of their taxonomies, or specialising branches that require more granularity. Given that the conceptualisation activity is implementation-language independent, it allows modeling ontologies according to the minimal encoding bias design criterion. The *formalisation* activity transforms the conceptual model into a formal or semi-computable model. The *implementation* activity builds computable models in an ontology language.

During the postdevelopment, the *maintenance* activity updates and corrects the ontology if needed. Also during the postdevelopment, the ontology is *(re)used* by other ontologies or applications. The evolution activity consists in managing ontology changes and their effects by creating and maintaining different variants of the ontology, taking into account that they can be used in different ontologies and applications (Noy et al., 2004).

Finally, *ontology support activities* include a series of activities that can be performed dur-

Figure 1. Ontology development process (adapted from Fernández-López et al., 1997)

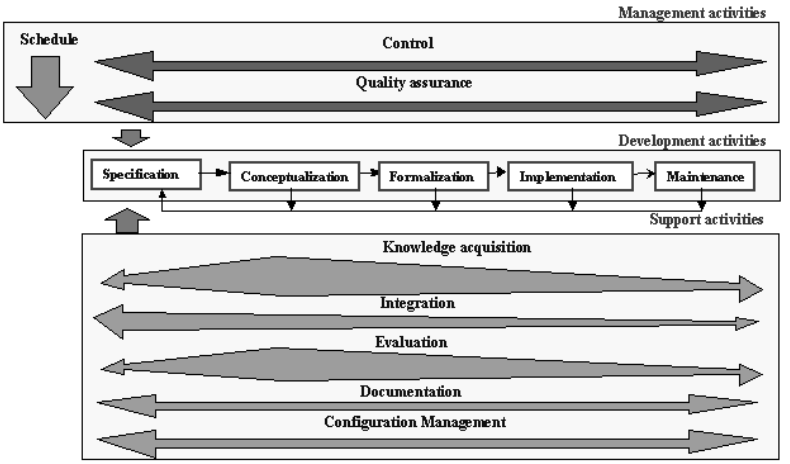


ing the development-oriented activities, without which the ontology could not be built. They include knowledge acquisition, evaluation, integration, merging, alignment, documentation, and configuration management. The goal of the *knowledge acquisition* activity is to acquire knowledge from experts of a given domain or through some kind of (semi)automatic process, which is called ontology learning (Kietz et al., 2000). The *evaluation* activity (Gómez-Pérez, 1994) makes a technical judgment of the ontologies, of their associated software environments, and of the documentation. This judgment is made with respect to a frame of reference during each stage and between stages of the ontology's lifecycle. The *integration* activity is required when building a new ontology by reusing other ontologies already available. Another support activity is *merging* (Gangemi et al., 1999; Noy & Musen, 2000; Steve et al., 1998; Stumme & Maedche, 2001), which consists in obtaining a new ontology starting from several ontologies on the same domain. The resulting ontology is able to unify concepts, terminology, definitions, constraints, and so forth, from all the source ontologies. The merge of two or more ontologies can be carried out either in run-time or design time. The *alignment* activity establishes different kinds of mappings (or links) between the involved ontologies. Hence this option preserves the original ontologies and does not merge them. The *documentation* activity details, clearly and exhaustively, each and every one of the completed stages and products generated. The *configuration management* activity records all the versions of the documentation and of the ontology code to control the changes. The *multilingualism activity* consists in mapping ontologies onto formal descriptions of linguistic knowledge (Declerck & Uszkoreit, 2003). It has not usually been considered as an ontology support activity, but has become more relevant in the context of networked ontologies available in the Semantic Web.

The ontology development process does not identify the order in which the activities should be performed. This is the role of the *ontology lifecycle*, which identifies *when* the activities should be carried out, that is, it identifies the *set of stages* through which the ontology moves during its life time, describes what activities are to be performed in each stage and how the stages are related (relation of precedence, return, etc.).

The initial version of the lifecycle process model of methontology (see Figure 2) proposes to start with a scheduling of the activities to be

Figure 2. Ontology lifecycle in methontology



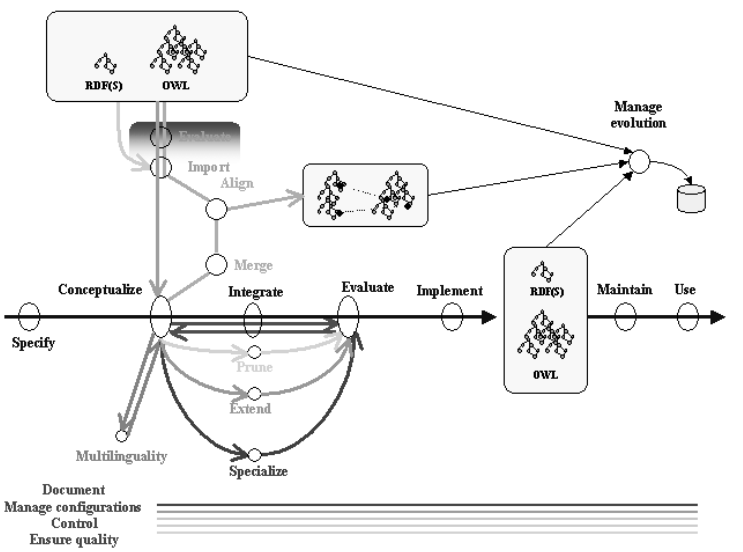
performed. Then, the specification activity begins, showing why the ontology will be built, which its possible uses will be, and who its users. When the specification finishes, the conceptualisation begins. The objective of the conceptualisation is to organise and structure the acquired knowledge in the knowledge acquisition activity, using a set of representations easy to manipulate for the experts on the domain. Once the conceptual model has been built, it has to be formalised and implemented (although if the conceptual model is formal enough then it will not be necessary to go through these two stages but just directly to the implementation). More details can be found in Gómez-Pérez et al. (2003).

The original ontology lifecycle of methontology has been modified recently to take into account the fact that more ontologies are available in ontology libraries or spread over the Internet, so that their reuse by other ontologies and applications has increased. Domain ontologies can be reused to build others of more granularity and coverage, or can be merged with others to create new ones. Using an analogy with an underground map, it can be noted that there exists a main line (in the middle of the Figure 3), which proposes the main development activities already identified in the early versions of methontology. Others lines start from the main one or finish in it, and others go in parallel ways and fork in a point. Thus, *interdependence relationships* (Gómez-Pérez & Rojas, 1999) arise between the lifecycle of several ontologies, and actions of evaluation, pruning and merging can be carried out on such ontologies. That is, the lifecycles of the different ontologies intersect, producing different scenarios with different technological requirements. Corcho and colleagues (2007) describe some of the most common scenarios that appear in this new context.

Methods and Methodologies Used for the Whole Ontology Development Lifecycle

Several methods and methodologies have been proposed in the literature as a guide for the main phases of the ontology development lifecycle. The

Figure 3. The ontology development process of networked ontologies



selection of one or another will mainly depend on the characteristics of the ontology to be developed, including the context where they are being developed and the experience of the ontology engineers with the each approach. At the end of this section we provide a comparison of the approaches according to several factors that can be considered for ontology development.

The **Cyc** method (Lenat & Guha, 1990), which is given this name because it was used for the development of the Cyc knowledge base, is mainly oriented to support the knowledge acquisition activity, and is structured in three phases. In all of them, the objective is to derive common sense knowledge that is implicit in different sources. The difference between them is the degree of automation of the knowledge acquisition process (from manual to automatic). Once knowledge has been acquired, it is divided into microtheories (or contexts), which are bundles of assertions in the same domain.

The *Uschold and King's* method (Uschold & King, 1995) covers more aspects of the ontology development lifecycle. It proposes four phases: (1) to identify the purpose of the ontology, (2) to build it, integrating other ontologies inside the current one if necessary, (3) to evaluate it, and (4) to document it. The authors 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, specialised into more specific concepts; a bottom-up approach, in which the most specific concepts are identified first and then generalised into more abstract concepts; and a middle-out approach, in which the most important concepts are identified first and then generalised and specialised into other concepts. Depending on the characteristics of the ontology to be developed, different strategies will be considered.

Grüninger and Fox (Grüninger & Fox, 1995) propose a methodology that is inspired on the development of knowledge-based systems using first order logic. They propose first to identify intuitively the possible applications where the ontology will be used, and determine the scope of the ontology using a set of natural language questions, called competency questions. These questions and their answers are used both to extract the main ontology components (expressed in first order logic). This methodology is very formal and can be used as a guide to transform informal scenarios in computable models.

In the method proposed in the *KACTUS* project (Bernaras et al., 1996) the ontology is built on the basis of an application knowledge base (KB), by means of a process of abstraction (that is, following a bottom-up strategy). The more applications are built, the more reusable and sharable the ontology becomes.

The method based on *Sensus* (Swartout et al., 1997) aims at promoting the sharability of knowledge, since it proposes to use the same base ontology to develop ontologies in particular domains. It is a top-down approach where 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 50,000 concepts). Then, all the concepts in the path from the seed terms to the ontology root are included. For those nodes that have a large number of paths through them, the entire subtree under the node is sometimes 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.

Methontology (Fernández-López et al., 1999) is a methodology that can be used to create domain ontologies that are independent of the application where they will be used. The ontology development process and lifecycle presented in the previous section are derived from this methodology. Besides, the methodology proposes specific techniques to carry out each of the activities identified there. The main phase in the ontology development process is the conceptualisation phase.

The *on-to-knowledge* methodology (Staab et al., 2001) is based on an analysis of usage scenarios. The steps proposed by the methodology are: *kick-off*, where ontology requirements are captured and 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*.

If we analyse the approaches according to the part of the ontology development process that they describe, we can conclude (Fernández-López & Gómez-Pérez, 2002a):

1. None of the approaches covers all the processes involved in ontology building. Most of the methods and methodologies for building ontologies are focused on the development activities, specially on the ontology conceptualisation and ontology implementation, and they do not pay too much attention to other important aspects related to management, learning, merge, integration, evolution and evaluation of ontologies. Therefore, such types of methods should be added to the methodologies for ontology construction from scratch (Fernández-López & Gómez-Pérez, 2002b).
2. Most of the approaches are focused on development activities, especially on the ontology implementation, and they do not pay too much attention to other important aspects related to the management, evolution and evaluation of ontologies. This is due to the fact that the ontological engineering field is relatively new. However, a low compliance with the criteria formerly established does not mean a low quality of the methodology or method. As de Hoog (1998) states, a not very specified method can be very useful for an experienced group.
3. Most of the approaches present some drawbacks in their use. Some of them have not been used by external groups and, in some cases they have been used in a single domain.
4. Most of the approaches do not have a specific tool that gives them technology support. Besides, none of the available tools covers all the activities necessary in ontology building.

Methods and Techniques Aimed at Specific Activities of the Ontology Development Process

Now we will provide an overview of some of the most important methods and techniques that are proposed to give support to specific activities of the ontology development process, such as those for ontology learning (which support the knowledge acquisition activity), ontology alignment and merge (which support the integration, merge and alignment activities), ontology evolution and versioning (which support the maintenance activity), and ontology evaluation.

Methods and Techniques for Ontology Learning

Ontology learning is defined as the set of methods and techniques used for building an ontology from scratch, enriching, or adapting an existing ontology in a semi-automatic fashion using distributed and heterogeneous knowledge and information sources, allowing to reduce the time and effort needed in the ontology development process. Though the fully automatic acquisition of knowledge remains far to be reached, the overall process is considered as semi-automatic, meaning that the human intervention is necessary in some parts of the learning process.

Several approaches have appeared during the

last decade for the partial automatisation of the knowledge acquisition process, applied to different types of unstructured, semistructured, and fully structured data (Maedche & Staab, 2000). Most of these approaches are based on linguistic patterns, which are used to extract linguistic relations that reflect ontological relations (taxonomic and nontaxonomic relations as well as possible attributes or their values, depending on the pattern's type). In the same sense, these patterns are also used for detecting attribute-value pairs. All the presented methods require the participation of an ontologist to evaluate the final ontology and the accuracy of the learning process. There are not methods or techniques for evaluating the accuracy of the learning process either.

Regarding *ontology learning methods*, some of the most known ones are due to Maedche and colleagues (Kietz et al., 2000), Aussenac-Gilles and colleagues (2000a, 2000b), and Khan and Luo (2002). *Maedche and colleagues' method* (Kietz et al., 2000) proposes to learn the ontology using as a base a core ontology (SENSUS, WordNet, etc.), which is enriched with the learnt concepts. New concepts are identified using natural language analysis techniques over the resources previously identified by the user. The resulting ontology is pruned and then focused on a specific domain by means of several approaches based on statistics. Finally, relations between concepts are established applying learning methods.

Aussenac-Gilles and colleagues' method (Aussenac-Gilles et al., 2000a, 2000b) combines knowledge acquisition tools based on linguistics with modeling techniques to keep links between models and texts. After selecting a corpus, the method proposes to obtain linguistic knowledge (terms, lexical relations, and groups of synonyms) at the linguistic level. This linguistic knowledge is then transformed into a semantic network, which includes concepts, relations and attributes.

Khan and Luo's method (Khan & Luo, 2002) aims to build a domain ontology from text documents using clustering techniques and WordNet (Miller, 1995). The user provides a selection of documents, which are clustered using the SOAT algorithm (Wu & Hsu, 2002). After building a hierarchy of clusters, a concept is assigned to each cluster in the hierarchy using a bottom-up fashion and a predefined set of topic categories. For this purpose, a topic tracking algorithm (Joachims, 1998) is used. Then, each topic is associated with an appropriate concept in WordNet, and other nodes in the hierarchy are assigned according to the concepts in the descendent nodes and their hyperyms in WordNet. Relations between concepts are ignored.

Methods and Techniques for Ontology Alignment and Merge

Ontologies aim to capture consensual knowledge of a given domain in a generic and formal way, to be reused and shared across applications and by groups of people. From this definition we could wrongly infer that there is only one ontology for modeling each domain (or even a single universal ontology). Though this can be the case in specific domains, commonly several ontologies model the same domain knowledge in different ways.

Noy and Musen (2000) defined ontology alignment and merging as follows: (1) *ontology alignment* consists in establishing different kinds of mappings (or links) between two ontologies, hence preserving the original ontologies (see Figure 4); and (2) *ontology merging* proposes to generate a unique ontology from the original ontologies. In this chapter we will assume that a *mapping* between ontologies is a set of rewriting rules that associates terms and expressions defined in a source ontology with terms and expressions of a target ontology (inspired from Mitra, Wiederhold, & Kersten, 2000). Table 1 shows the mappings that can be established between the two ontologies of Figure 4. The symbol ":=" means "is transformed into," and "λ" is the empty word. Therefore, `date := λ` means that the attribute

date has no correspondence with terms of the ontology 2.

Given that a reusable and machine interpretable database schema can be considered as an ontology (see second section), the galaxy of ontology alignment methods is huge. Some examples of these methods are: *S-Match* (Shvaiko et al., 2004), *QOM* (Ehring & Staab, 2004), *Pan and colleagues proposal* (2005), *Artemis* (Benebentano et al., 2000; Castano et al., 2001), *Cupid* (Madhavan et al., 2001), *AnchorPrompt* (Noy & Musen, 2001), *Similarity Flooding* (Melnik et al., 2002), and so forth.

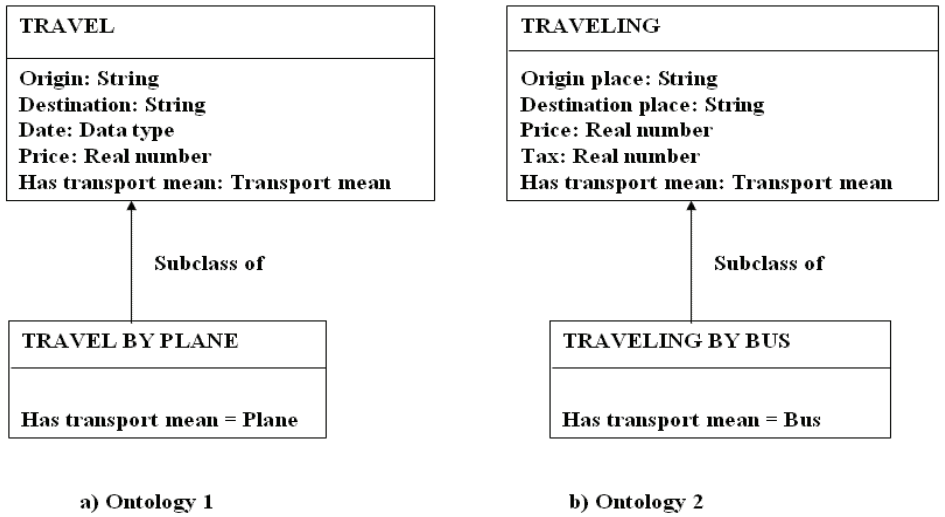
In the context of the workshop on Evaluation of Ontology Tools EON2004, an experiment was performed about the quality of the mappings provided by different methods and tools. This will be continued in other efforts. To know more on ontology alignment and merging we recommend to access to the Ontology Matching Web page.[4]

With regard to *ontology merging methods and methodologies*, one of the most elaborated proposals for ontology merging is *ONIONS* (Gangemi et al., 1999; Steve et al., 1998), developed by the Conceptual Modeling Group of the CNR in Rome, Italy. With this method we can create a library of ontologies originated from different sources. The main underlying ideas of this method are: (1) to link the ontologies taking into account lexical relations between their terms (polysemy, synonymy, etc.); and (2) to use generic theories (part-whole or connectedness theories, for example) as common upper ontologies of the library ontologies, that is,

Table 1. Mappings for the two ontologies of Figure 4

Description of the mapping in natural language	Rewriting rule
The concept travel (in ontology 1) is equivalent to the concept traveling (in ontology 2).	Travel := Traveling
The concept travel by plane (in ontology 1) is equivalent to the concept such as it is subclass of traveling (in ontology 2) and its transport mean is a plane (in ontology 2).	TravelByPlane := C such as subclassOf(C, Traveling) ∧ C.hasTransporMean = Plane
The concept such as it is subclass of travel (in ontology 1) and its transport mean is a `bus` (in ontology 2) is equivalent to the concept traveling by bus (in ontology 2).	C such as subclassOf(C, Travel) ∧ C.hasTransporMean = Bus := TravelingByBus
The attribute origin (in ontology 1) is equivalent to the attribute origin place (in ontology 2).	Origin := OriginPlace
The attribute destination (in ontology 1) is equivalent to the attribute destination place (in ontology 2).	Destination := DestinationPlace
The value New York of attributes origin and destination (in ontology 2) is equivalent to the value NY of origin place and destination place (in ontology 2).	"New York" := "NY"
The attribute date (in ontology 1) does not have correspondence in ontology 2.	Date := λ
The attribute price (in ontology 1) is equivalent to a combination of the attributes price and tax in ontology 2.	Price := Price * (1 + Tax/100)
The attribute `has` transport mean (in ontology 1) is equivalent to the attribute `has` transport mean in ontology 2.	HasTransportMean := HasTransportMean

Figure 4. Example of ontology alignment



to use generic theories as the glue to integrate the different ontologies.

FCA-Merge (Stumme & Maedche, 2001) was developed at the Institute AIFB of the University of Karlsruhe, Germany. This approach is very different from the other approaches presented in this section. FCA-Merge takes as input the two ontologies to be merged and a set of documents on the domains of the ontologies. The appearances of instances of the concepts in the different documents guides the merging such concepts.

The *PROMPT* method (Noy & Musen, 2000) has been elaborated by the Stanford Medical Informatics group at Stanford University. The main assumption of PROMPT is that the ontologies to be merged are formalised with a common knowledge model based on frames. This method proposes first to elaborate a list with the candidate operations to be performed to merge the two ontologies (e.g., merge two classes, merge two slots, etc.). Afterwards, a cyclic process starts. In each cycle the ontologist selects an operation of the list and executes it.

PromptDiff is a component of Prompt (Noy & Musen, 2004b) that allows maintaining ontology views or mappings between ontologies. PromptDiff provides an ontology-comparison API that other applications can use to determine, for example, the mapping needs to be updated when new versions of mapped ontologies appear (Noy et al., 2004).

Methods and Techniques for Ontology Evolution and Versioning

Ontologies are often developed by several groups of people and may evolve over time. Therefore, they cannot be understood as static entities, but rather as dynamic ones. As a consequence, ontology versioning becomes necessary and essential.

Ontology engineers working in parallel on the same ontology need to maintain and compare different versions, to examine the changes that others have performed, and to accept or reject the changes. Ontology-based application developers should easily see the changes between ontology versions, determine which definitions were added or deleted, and accept or reject the changes. Let's note that, for ontologies, we must compare the

semantics of the ontologies and not their serialisations, since two ontologies that are exactly the same conceptually may have very different text representations when implemented in some ontology languages.

The most relevant methods (and corresponding tools) for ontology evolution and versioning are the *change management KAON plug-in* (Stojanovic, 2004) and the *PromptDiff algorithm* (Noy et al., 2004).

The *change management KAON plug-in* allows establishing the effects of changes through *evolution strategies* (Stojanovic, 2004). A particular evolution strategy allows establishing, for example, what happens with its subclasses when a concept *C* is deleted: if they can be also deleted, or they can become subclasses of the superclasses of *C*.

The *PromptDiff algorithm*, which is integrated in the PROMPT plug-in of the Protégé tool, compares ontologies producing an initial set of mappings between two versions of the same ontology (Noy et al., 2004). For instance, if a term t_1 of the version v_1 has the same type as the term t_2 of the version v_2 (both of them are concepts, both of them are properties, etc.) and t_1 has a similar name to t_2, it is assumed that the semantics of t_1 and t_2 are similar. Therefore, t_1 and t_2 are mapped as similar terms. This initial set of mappings is propagated using a fixed-point algorithm that combines the results of the previous step. Thus, for example, if all the siblings of the concept c_1 of v_1 are mapped with siblings of the concept c_2 of v_2, c_1 and c_2 are candidates to be mapped through a change operation (e.g., the addition of a new subclass).

Methods and Techniques for Ontology Evaluation

Work on ontology content evaluation was started by Gómez-Pérez (1994). A survey on evaluation methods and tools can be found in (Gómez-Pérez et al., 2003). These evaluation efforts can be examined under the following four perspectives:

From a *content perspective*, many libraries exist where ontologies are published and publicly available (SWOOGLE[5], Oyster[6], DAML[7], Protégé[8], etc.). No documentation is available about how ontologies available in libraries or well-known and large ontologies (e.g., Cyc (Lenat & Guha 1990), or Sensus (Swartout et al., 1997)) were evaluated. However they have been used to build many successful applications.

From a *methodology perspective*, the main efforts to evaluate ontology content were made by Gómez-Pérez (1996, 2001) in the framework of methontology, and by Guarino and colleagues (Welty & Guarino, 2001) with the OntoClean method.

Gómez-Pérez has identified and classified different kinds of errors in taxonomies. Such identification can be used as a checklist for taxonomy evaluation. Such a list presents a set of possible errors that can be made by ontology engineers when modeling taxonomic knowledge in an ontology under a frame-based approach. They are classified in: inconsistency, incompleteness, and redundancy errors. The ontology engineer should not postpone the evaluation until the taxonomy is finished; the control mechanisms should be performed during the construction of the taxonomy.

OntoClean is a method elaborated by the Ontology Group of the CNR in Padova (Italy). Its goal is to remove wrong *Subclass-Of* relations in taxonomies according to some philosophical notions such as *rigidity*, *identity* and *unity*. According to this method, the ontology engineer, first, assigns some meta-properties to each concept of the taxonomy (for example, if each instance of the concept is a whole), then it applies a set of rules that establish the possible incompatibilities of values in the taxonomy. Such rules allow pruning wrong *subclass of* links if the values assigned to a concept are incompatible with the values assigned to its children.

Recently, some researchers have published a synthesis of their experience in ontology evaluation (Daelemans & Reinberger, 2004; Gómez-Pérez, 2004; Guarino, 2004; Noy, 2004). According to their conclusions, although good ideas have been provided in this area, there are still important lacks. Other interesting works are (Guo, Pan, & Heflin, 2004) and the aforementioned EON2004 experiment.

ONTOLOGY TOOLS

Ontology tools appeared in the mid-1990s with the objective of giving support to the development of ontologies, either following a specific set of methods or a methodology or not. Taking into account the characteristics of their knowledge models, ontology tools can be classified in the following two groups:

- Tools whose knowledge model maps directly to an ontology language, hence developed as ontology editors for that specific language. This groups includes: the Ontolingua Server (Farquhar, Fikes, & Rice, 1997), which supports ontology construction with Ontolingua and KIF; OntoSaurus (Swartout et al., 1997) with Loom; WebOnto (Domingue, 1998) with OCML; OilEd (Bechhofer, Horrocks, Goble, & Stevens, 2001) with OIL first, later with DAML+OIL, and finally with OWL; and SWOOP (Kalyanpur, Parsia, & Hendler, 2005) and KAON2 (Hustadt, Motik, & Sattler, 2004) with OWL.
- Integrated tool suites whose main characteristic is that they have an extensible architecture, and whose knowledge model is usually independent of ontology languages. These tools provide a core set of ontology related services and are easily extended with other modules to provide more functions. In this group we have included Protégé (Noy, Fergerson, & Musen, 2000), WebODE (Arpírez, Corcho, Fernández-López, Gómez-Pérez, 2003; Corcho, Fernández-López, Gómez-Pérez, & Vicente, 2002), OntoEdit (Sure, Erdmann, Angele, Staab, Studer, & Wenke, 2002), and KAON1 (Maedche, Motik, Stojanovic, Studer, & Volz, 2003).

Tools that Give Support to Most of the Activities of the Ontology Development Process

In this section we will focus on those tools that give an integrated support to the ontology development process, and consequently cover most of the activities needed to develop ontologies. From all of them we will only describe those that belong to the new generation of ontology-engineering environments, in particular, in Protégé, WebODE, OntoEdit and KAON1.[9]

These tools 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.

Protégé (Noy et al., 2000) has been developed by the Stanford Medical Informatics (SMI) at Stanford University. 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 import/export (FLogic, Jess, XML, Prolog), ontology language design (Knublauch, Fergerson, Noy, & Musen, 2004), OKBC access, constraints creation and execution (PAL), ontology merge (Prompt (Noy & Musen, 2000)), and so forth.

WebODE (Arpírez et al., 2003; Corcho et al., 2002) has been developed by the Ontological Engineering Group of 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 several frontends. 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 import/export (XML, RDF(S), OWL, CARIN, FLogic, Jess, Prolog), axiom edition, 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 (Sure et al., 2002) has been developed by AIFB in Karlsruhe University, and is commercialised by Ontoprise. 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, of exporting and importing ontologies in different formats (FLogic, XML, RDF(S) and OWL), and so forth. Two versions of OntoEdit are available: OntoEdit Free and OntoEdit Professional.

The *KAON1* tool suite (Maedche et al., 2003) is an open source extensible ontology engineering environment. The core of this tool suite is the ontology API, which defines its underlying knowledge model based on an extension of RDF(S). The OI-modeler is the ontology editor of the tool suite that provides capabilities for ontology evolution, ontology mapping, ontology generation from databases, and so forth.

An interesting aspect of tools is that only OntoEdit and WebODE give *support to ontology building methodologies* (on-to-knowledge and methontology respectively), though this does not prevent them from being used with other methodologies or with no methodology at all.

From the *KR paradigm* point of view, KAON is based on semantic networks plus frames, and the rest of tools 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, relations, attributes, and instances. Only KAON1, and Protégé provide flexible modeling 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. KAON1 does not have an inference engine. OntoEdit uses FLogic (Kifer, Lausen, & Wu, 1995) as its inference engine, WebODE uses Ciao Prolog (Hermenegildo, Bueno, Cabeza, Carro, García, López, & Puebla, 2000), and Protégé uses an internal PAL engine. Besides, Protégé and WebODE provide ontology evaluation facilities. WebODE and Protégé include a module that performs ontology evaluation according to the OntoClean method (Guarino & Welty, 2002; Welty & Guarino, 2001). Finally, Protégé (with the OWL plug-in) performs automatic classifications by means of connecting to a description logic reasoner.

Another important aspect to take into account in ontology tools 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, all

these tools are based on Java platforms and provides database storage support. Backup management functionality is just provided by WebODE, and extensibility facilities are allowed in KAON, OntoEdit, Protégé and WebODE.

Related to the *cooperative and collaborative construction of ontologies*, Protégé incorporates some synchronisation functionalities. In general, more features are required in existing tools to ensure a successful collaborative building of ontologies.

Tools that Give Support to Specific Activities of the Ontology Development Process

Here we will only cover tools for ontology learning and ontology merge and alignment, since the ones for evolution and evaluation are very close to each of the methods described previously and consequently there is not much more that can be described about them.

Tools that Give Support to Ontology Learning

We will describe Caméléon (Aussenac-Gilles & Seguela, 2000), LTG Text Processing Workbench (Mikheev & Finch, 1997), Prométhée (Morin, 1998, 1999), SOAT tool (Wu & Hsu, 2002) and Text-To-Onto (Maedche & Staab, 2000).

Caméléon (Aussenac-Gilles & Seguela, 2000) assists in learning conceptual relations to enrich conceptual models. Caméléon relies on linguistic principles for relation identification: lexico-syntactic patterns are good indicators of semantic relations. Some patterns may be regular enough to indicate the same kind of relation from one domain to another. Other patterns are domain specific and may reveal domain specific relations. This tool gives technological support to some steps of the Aussenac-Gilles and colleagues' method.

Language Technology Group (*LTG*) *Text Processing Workbench* (Mikheev & Finch, 1997) is a set of computational tools for uncovering internal structure in natural language texts written in English. The main idea behind the workbench is the independence of the text representation and text analysis. In LTG, ontology learning is performed in two sequential steps: representation and analysis. At the representation step, the text is converted from a sequence of characters to features of interest by means of annotation tools. At the analysis step, those features are used by tools of statistics-gathering and inference to find significant correlations in the texts. The workbench is being used both for lexicographic purposes and for statistical language modeling.

Prométhée (Morin, 1998, 1999) is a machine learning based tool for extracting and refining lexical-syntactic patterns related to conceptual specific relations from technical corpora. It uses pattern bases, which are enriched with the ones extracted in the learning. To refine patterns, the authors propose the Eagle (Guarino, Masolo, & Vetere, 1999) learning system. This system is based on the inductive paradigm *learning from examples*, which consists in the extraction of intentional descriptions of target concepts from their extensional descriptions, and previous knowledge on the given domain. This fact specifies general information, like the object characteristics and their relations. The tool extracts *intentional* descriptions of concepts from their *extensional* descriptions. The learned definitions are later used in recognition and classification tasks.

SOAT (Wu & Hsu, 2002) allows a semi-automatic domain ontology acquisition from a domain corpus. The main objective of the tool is to extract relationships from parsed sentences based on applying phrase-rules to identify keywords with strong semantic links like hyperonyms or synonyms. The acquisition process integrates linguistic, commonsense, and domain knowledge. The restrictions of SOAT involve that the quality of the corpus must be very high, in the sense that the sentences must be accurate and enough

to include most of the important relationships to be extracted.

Text-To-Onto (Maedche & Staab, 2000) integrates an environment for building domain ontologies from an initial core ontology. It also discovers conceptual structures from different German sources using knowledge acquisition and machine learning techniques. Text-To-Onto has implemented some techniques for ontology learning from free and semistructured text. The result of the learning process is a domain ontology that contains domain-specific and domain-independent concepts. Domain-independent concepts are withdrawn to better adjust the vocabulary of the domain ontology. The result of this process is a domain ontology that only contains domain concepts learnt from the input sources related before. The ontologist supervises the whole process. This is a cyclic process, in the sense that it is possible to refine and complete the ontology if we repeat the process.

An important conclusion that we can obtain in the revision of ontology learning tools is that it does not exist a fully automatic tool that carries out the learning process. Some tools are focused on helping in the acquisition of lexico-semantic knowledge, others help to elicit concepts or relations from a pre-processed corpus with the help of the user, and so forth. A deeper description of methods and tools can be found in (Gómez-Pérez & Manzano, 2003).

Tools that Give Support to Ontology Alignment and Merge

With regard to ontology alignment tools, we will describe the QOM toolset, S-Match, Pan and colleagues tool and OLA.

The *QOM toolset* (Ehring & Staab, 2004) gives support to the QOM method. It is implemented in Java using the KAON framework. It has been basically used to make experiments with the method and compare it with other methods.

S-Match tool translates and preprocesses the input ontologies. Then, it orders the transformation of prefixes, the expansions of abbreviations, and so forth. Later, using resources like Wordnet, it generates a first mapping base. Finally, using the SAT solvers, new mappings are generated.

Pan and colleagues (2005) apply their method combining the Google search engine and text classifiers (such as Rainbow[10] or cbacl[11]) to calculate the prior probabilities of the Bayesian network. Then, the subsequent probability is calculated using any Bayesian network tool.

OLA[12] (Euzenat, 2004) is an API for manipulating alignments between ontologies in OWL. It allows applying and combining different algorithms, and even adding others new. Currently, this API has been mainly used with mapping methods based on lexical similarity measures. OLA implements a format for expressing alignments in RDF.

With regard to ontology merge tools, we will describe OGSERVER, Chimaera, the Prompt plug-in, the FCA-Merge toolset and GLUE.

OBSERVER (Mena, Kashyap, Sheth, & Illarramendi, 1996) merges automatically ontologies of the same domain to access heterogeneous information sources. However, the merge process is carried out by an internal module and, therefore, it is invisible to the user.

Chimaera (McGuinness, Fikes, Rice, & Wilder, 2000) was built by the Knowledge Systems Laboratory (KSL) to aid in the process of ontology merge, and the *Prompt plug-in* (Noy & Musen, 2000), integrated in Protégé, was built by the Stanford Medical Informatics (SMI). The added value of the latter was that it provided support to the ontology merge method Prompt.

Approximately at the same time, the Institute AIFB of the University of Karlsruhe developed the *FCA-Merge toolset* (Stumme & Maedche, 2001) to support the FCA-Merge method.

Finally, in 2002, *GLUE* (Doan, Madhavan, Domingos, & Halevy, 2002) was developed at the University of Washington. GLUE is a system

that semi-automatically finds mappings between concepts from two different ontologies.

The current ontology merging approaches have the following lacks: (1) mappings to perform the merging are usually established by hand; (2) all the tools need the participation of the user to obtain the definitive result of the merging process; and (3) no tool allows merging axioms and rules. The natural evolution of merging tools should lead to increase the use of knowledge and to decrease the participation of the people in the process. This could improve the possibilities of the merging at run-time.

ONTOLOGY LANGUAGES

Ontology languages started to be created at the beginning of the 1990s, normally as the evolution of existing knowledge representation (KR) languages. Basically, the KR paradigms underlying such ontology languages were based on first order logic (e.g., KIF (Genesereth & Fikes, 1992)), on frames combined with first order logic (e.g., Ontolingua (Farquhar et al., 1997) (Gruber, 1992), OCML (Motta, 1999) and FLogic (Kifer et al., 1995)), and on description logics (e.g., Loom (MacGregor, 1991)). In 1997, OKBC (Chaudri et al., 1998) was created as a unifying frame-based protocol to access ontologies implemented in different languages (Ontolingua, Loom and CycL, among others). However it was only used in a small number of applications.

The boom of the Internet led to the creation of ontology languages for exploiting the characteristics of the Web. Such languages are usually called *Web-based ontology languages* or *ontology markup languages*. Their syntax is based on existing markup languages such as HTML (Raggett. , Le Hors, & Jacobs, 1999) and XML (Bray, Paoli, Sperberg-McQueen, & Maler, 2000), whose purpose is not ontology development but data presentation and data exchange respectively. The most important examples of these markup languages are: SHOE (Luke & Helfin, 2000), XOL (Karp, Chaudhri, & Thomere, 1999), RDF (Lassila & Swick, 1999), RDF Schema (Brickley & Guha, 2004), OIL (Horrocks, Fensel, Harmelen, Decker, Erdmann, & Klein, 2000), DAML+OIL (Horrocks & van Harmelen, 2001), and OWL (Dean & Schreiber, 2004). From all of them, the ones that are being actively supported are now RDF, RDF Schema and OWL. Finally, in the context of the work on Semantic Web Services and more specifically in the context of the WSMO framework, a new ontology language is being developed, named WSML.

We will describe the most relevant ontology mark-up languages, since they are the most useful for the work on Semantic Web Services.

RDF (Lassila & Swick, 1999) was developed by the W3C (the World Wide Web Consortium) as a semantic-network based language to describe Web resources. Finally, the RDF Schema (Brickley & Guha, 2004) language was also 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) only allows the representation of concepts, taxonomies of concepts and binary relations. Some inference engines and query languages have been created for this language.

Ontology Web Language (OWL) was proposed as a W3C recommendation in February 2004. OWL is built on top of RDF(S), extending its expressiveness with more primitives that allow representing complex expressions to describe concepts and relations. OWL is divided into three layers (OWL Lite, OWL DL and OWL Full), each of them providing different levels of expressiveness that can be used depending on the representation and inference needs of an ontology. OWL is based on the description logic language SHOIN(D+) and has several inference engines that can be used for constraint checking of concepts, properties and instances, and for automatic classification of concepts into hierarchies.

For instance, using OWL we can describe a

flight as a kind of travel where the means of transport used is a plane. If we specify this condition as necessary and sufficient and then we define a travel where a light aircraft is used as the means of transport (and we assume that light aircraft is a specialisation of a plane) then a reasoner will be able to derive that this travel is a specialisation of a flight. Similarly, this same principle can be used for checking the consistency of the definitions provided in an ontology.

Finally, *Web Service Modeling Language* (WSML) (de Bruijn, 2006) is being developed in the context of the WSMO framework.[13] This language is aimed to be used not only for representing ontologies, but also for representing Semantic Web Services; hence it contains many additional features that are not present in the languages aforementioned. Like OWL, it is divided in several layers. Each of these layers is based on different KR formalisms: description logic, logic programming and first order logic.

CONCLUSION

In the beginning of the 1990s ontology development was similar to an art: ontology developers did not have clear guidelines on how to build ontologies but only some design criteria to be followed. Work on principles, methods and methodologies, together with supporting technology, made ontology development become an engineering. This migration process was mainly due to the definition of the ontology development process and the ontology lifecycle, which described the steps to be performed in order to build ontologies and the interdependencies among all those steps.

In this chapter we have reviewed existing ontology principles, methods and methodologies, tools, and languages. The following is a summary of the chapter:

Ontology engineers have available methodologies that guide them along the ontology development process. Methontology is the methodology that provides the most detailed descriptions of the processes to be performed; On-To-Knowledge is the one that covers most activities, although with very short descriptions of processes; and Grüninger and Fox methodology is the most formal one. All of them consider the reuse of existing ontologies during the development process, but only methontology has recently adapted its proposal for a lifecycle to the environment of networked ontologies. In any case, the development activities are the most detailed in all of them, mainly the specification, conceptualisation and implementation. There is still a lack of proposals for ontology management activities (scheduling, control and quality assurance), and for some pre-development (e.g., environment study) and post-development activities (e.g., (re)use).

Concerning support activities, some interesting contributions have been done in ontology learning, ontology merging and alignment, ontology evolution, and ontology evaluation. Nevertheless, important work has to be done in all of these activities. For example, the time in which activities like ontology learning or ontology merging are applied to heavyweight ontologies is far away.

One of the problems that the ontology engineer can find when (s)he has to build an ontology is that (s)he has to use different methods that are not integrated. For example, ontology learning methods are not integrated in methodologies that cover the whole development process (e.g., in methontology or On-To-Knowledge). Some experiences exist in the integration of methods in methodologies. For example, the OntoClean method has been integrated in methontology (Fernández-López & Gómez-Pérez, 2002b).

A similar problem appears in the use of ontology tools, given that there is a lack of integrated environments for ontology development. Tools are usually created as isolated modules that solve one type of problems, but neither are fully integrated nor do they interoperate with other tools that implement other activities of the ontology

lifecycle.

Finally, work on ontology languages has been in constant evolution since the first languages that were made available for ontology implementation, most of them based on existing knowledge representation languages. The existence of heterogeneous networked ontologies has been mainly considered in the recent language developments created in the context of the Semantic Web (RDF, RDF Schema and OWL) and of Semantic Web Services (WSML), with the addition of namespaces that allow referring to ontology components that have been defined elsewhere and with the use of import primitives to include an existing model in an ontology.

ACKNOWLEDGMENTS

This work has been partially supported by the IST project Knowledgeweb (FP6-507482) and by the Spanish project Semantic Services (TIN 2004-02660).

REFERENCES

Arpírez, J.C., Corcho, O., Fernández-López, M., & Gómez-Pérez, A. (2003). WebODE in a nutshell. *AI Magazine*.

Aussenac-Gilles, N., Biébow, B., & Szulman, S. (2000a). Revisiting ontology design: A methodology based on corpus analysis. In R. Dieng & O. Corby (Eds.), *Proceedings of the 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00),* Juan-Les-Pins, France, (*LNAI,* 1937 , pp. 172-188). Berlin: Springer-Verlag.

Aussenac-Gilles, N., Biébow, B., & Szulman, S. (2000b). Corpus analysis for conceptual modelling. In N. Aussenac-Gilles, B. Biébow & S. Szulman (Eds.), *Proceedings 51 of EKAW'00 Workshop on Ontologies and Texts,* Juan-Les-Pins, France. (pp. 1.1-1.8), CEUR Workshop. Amsterdam, The Netherlands. Retrieved October 23, 2006, from http://CEUR-WS.org/Vol-51/

Aussenac-Gilles, N. & Seguela, P. (2000). Les relations sémantiques: du linguistique au formel. In A. Condamines (Ed.), *Cahiers de grammaire, N° spécial sur la linguistique de corpus* (Presse de l'UTM, Vol 25, pp. 175-198). Toulouse.

Bechhofer, S., Horrocks, I., Goble, C., & Stevens, R. (2001). OilEd: A reasonable ontology editor for the Semantic Web. In F. Baader, G. Brewka, & T. Eiter (Eds.), *Joint German/Austrian conference on Artificial Intelligence (KI'01)* (pp. 396-408), Vienna, Austria. Lecture Notes in Artificial Intelligence 2174. Berlin: Springer-Verlag.

Beneventano, D., Bergamaschi, S., Castano, S., Corni, A., Guidetti, R., Malvezzi, G., Melchiori, M., & Vincini, M. (2000). Information integration: The MOMIS project demonstration. In A. El Abbadi, M.L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, & K.Y. Whang (Eds.), *26th International Conference On Very Large Data Bases* (pp. 611-614), El Cairo, Egypt. San Francisco: Morgan Kaufmann Publishers.

Bernaras, A., Laresgoiti, I., & Corera, J. (1996). Building and reusing ontologies for electrical network applications. In W. Wahlster (Ed.), *European Conference on Artificial Intelligence (ECAI'96),* Budapest, Hungary, (pp. 298-302). Chichester, UK: John Wiley and Sons.

Berners-Lee, T. (1999). *Weaving the Web: The original design and ultimate destiny of the World Wide Web by itsiInventor.* New York: HarperCollins Publishers.

Bray, T., Paoli, J., Sperberg-McQueen, C.M., & Maler, E. (2000). Extensible markup language (XML) 1.0. W3C Recommendation. Retrieved October 23, 2006, from http://www.w3.org/TR/REC-xml

Brickley, D., & Guha, R.V. (2004). *RDF* vocabulary description language 1.0: RDF schema. W3C Recommendation. Retrieved October 23, 2006, from http://www.w3.org/TR/PR-rdf-schema

Castano, S., De Antonellis, V., & De Capitani diVemercati, S. (2001). *Global viewing of heterogeneous data sources. IEEE Transactions on Data Knowledge Engineering, 13*(2), 277–297.

Corcho, O., Fernández-López, M., Gómez-Pérez, A., & Vicente, O. (2002). 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)* (pp. 138-153), Sigüenza, Spain. Lecture Notes in Artificial Intelligence 2473. Berlin: Springer-Verlag.

Corcho, O., Fernández-López, M., & Gómez-Pérez, A. (2007). Ontological engineering: Principles, methods, tools and languages. In C. Calero, F. Ruiz, & M. Piattini (Eds.), *Ontologies for Software Engineering and Technology*. Springer-Verlag.

Chaudhri, V.K., Farquhar, A., Fikes, R., Karp, P.D., & Rice, J.P. (1998). *Open knowledge base connectivity 2.0.3* (Tech. Rep.). Retrieved October 23, 2006, from http://www.ai.sri.com/~okbc/okbc-2-0-3.pdf

Chen, P.P. (1976). The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems, 1*(1), 9–36.

Daelemans, W. & Reinberger, M.L. (2004). Shallow text understanding for ontology content evaluation. *IEEE Intelligence Systems, 19*(4), 76-78.

De Bruijn, J. (2006). *The Web service modeling language WSML* (Deliverable D16.1v0.21). Retrieved October 23, 2006, from http://www.wsmo.org/TR/d16/d16.1/v0.21/

de Hoog, R. (1998). Methodologies for building knowledge based systems: Achievements and prospects. In J. Liebowitz (Ed.), *Handbook of expert systems (Chapter 1)*. Boca Raton, FL: CRC Press.

Dean, M. & Schreiber, G. (2004). *OWL Web ontology language reference*. W3C Recommendation. Retrieved October 23, 2006, from http://www.w3.org/TR/owl-ref/

Declerck, T. & Uszkoreit, H. (2003). State of the art on multilinguality for ontologies, annotation services and user interfaces. Esperonto deliverable D1.5. Retrieved October 22, 2006, from http://www.esperonto.net

Doan, A., Madhavan, J., Domingos, P., & Halevy, A. (2002). Learning to map between ontologies on the Semantic Web. In D. Lassner (Ed.), *Proceedings of the 11th International World Wide Web Conference (WWW 2002)*, Honolulu, Hawaii. Retrieved October 23, 2006, from http://www2002.org/refereedtrack.html

Domingue, J. (1998). Tadzebao and webOnto: Discussing, browsing, and editing ontologies on the Web. In B.R. Gaines & M.A. Musen (Eds.), *11th International Workshop on Knowledge Acquisition, Modeling and Management (KAW'98)* (KM4, pp. 1-20), Banff, Canada.

Ehring, M., & Staab, S. (2004). QOM – quick ontology mapping. In S.A. McIlraith & D. Plexousakis (Eds.), *3rd International Semantic Web Conference (ISWC'04),* Hiroshima, Japan, (*LNCS* 3298, pp. 683-697). Berlin: Springer-Verlag.

Euzenat, J. (2004). An API for ontology alignment. In S.A. McIlraith & D. Plexousakis (Eds.), *3rd International Semantic Web Conference (ISWC'04)*, Hiroshima, Japan, (*LNCS* 3298, pp. 698-712). Berlin: Springer-Verlag.

Farquhar, A., Fikes, R., & Rice, J. (1997). The ontolingua server: A tool for collaborative ontology construction. *International Journal of Human Computer Studies, 46*(6), 707–727.

Fernández-López, M., & Gómez-Pérez, A. (2002a). Overview and analysis of methodologies for building ontologies. *The Knowledge Engineering Review, 17*(2), 129-156.

Fernández-López, M. & Gómez-Pérez, A. (2002b). The integration of ontoClean in webODE. In J. Angele & Y. Sure (Eds.), *Proceedings 62 of CEUR Workshop EKAW'02 Workshop on Evaluation of Ontology-based Tools (EON2002),* Sigüenza, Spain, (pp. 38-52). Amsterdam, The Netherlands. Retrieved October 23, 2006, from http://CEUR-WS.org/Vol-62/

Fernández-López, M., Gómez-Pérez, A., & Juristo, N. (1997). *Methontology: From ontological art towards ontological engineering*. Paper presented at the Spring Symposium on Ontological Engineering of AAAI (pp. 33-40). Stanford University.

Fernández-López, M., Gómez-Pérez, A., Pazos, A., & Pazos, J. (1999). Building a chemical ontology using methontology and the ontology design environment. *IEEE Intelligent Systems & Their Applications, 4*(1), 37-46.

Gangemi, A., Pisanelli, D.M., & Steve, G. (1999). An overview of the ONIONS project: Applying ontologies to the integration of medical terminologies. *Data & Knowledge Engineering, 31*(2), 183–220.

Genesereth, M.R. & Fikes, R.E. (1992). *Knowledge interchange format. Version 3.0. Reference Manual* (Tech. Rep. Logic-92-1). Stanford University, Computer Science Department. Retrieved October 23, 2006, from http://meta2.stanford.edu/kif/Hypertext/kif-manual.html

Gómez-Pérez, A. (2004). Evaluating ontology evaluation. *IEEE Intelligence Systems, 19*(4), 74-76.

Gómez-Pérez, A. (1994). *Some ideas and examples to evaluate ontologies*. Stanford University, Knowledge Systems Laboratory. Retrieved October 23, 2006, from http://www-ksl.stanford.edu/KSL_Abstracts/KSL-94-65.html

Gómez-Pérez, A. (1996). A framework to verify knowledge sharing technology. *Expert Systems with Application, 11*(4), 519-529.

Gómez-Pérez, A. (2001). Evaluation of ontologies. *International Journal of Intelligent Systems, 16*(3), 391-409.

Gómez-Pérez, A., Fernández-López, M., & Corcho, O. (2003). *Ontological engineering.* London: Springer.

Gómez-Pérez, A., Fernández-López, M., & de Vicente, A. (1996). *Towards a method to conceptualize domain ontologies*. In P. van der Vet (Ed.), *ECAI'96 Workshop on Ontological Engineering* (pp. 41-52), Budapest, Hungary.

Gómez-Pérez, A., & Manzano, D. (2003). A survey of ontology learning methods and techniques. OntoWeb deliverable D.1.5. Retrieved October 23, 2006, from http://www.ontoweb.org

Gómez-Pérez, A. & Rojas, M.D. (1999). Ontological reengineering and reuse. In D. Fensel & R. Studer (Eds.), *11th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW'99),* Dagstuhl Castle, Germany, (*LNAI* 1621, pp. 139-156). Berlin: Springer-Verlag.

Gruber, T.R. (1992). *Ontolingua: A mechanism to support portable ontologies* (Tech. Rep. No. KSL-91-66). Stanford University, Knowledge Systems Laboratory, Stanford, California. Retrieved October 23, 2006, from *f*tp://ftp.ksl.stanford.edu/pub/KSL_Reports/KSL-91-66.ps

Gruber, T.R. (1993). A translation approach to portable ontology specification. *Knowledge Acquisition, 5*(2), 199-220.

Grüninger, M. & Fox, M.S. (1995). Methodology for the design and evaluation of ontologies. In D. Skuce (Ed.), *IJCAI95 Workshop on Basic*

Ontological Issues in Knowledge Sharing (pp. 6.1–6.10).

Guarino, N. (2004). Toward formal evaluation of ontology quality. *IEEE Intelligence Systems, 19*(4), 78-79.

Guarino, N., Masolo, C., & Vetere, G. (1999). OntoSeek: Content-based access to the Web. *IEEE Intelligent Systems & Their Applications, 14*(3), 70–80.

Guarino, N., & Welty, C. (2002). Evaluating ontological decisions with ontoClean. *Communications of the ACM, 45*(2), 61-65.

Guo, Y., Pan, Z., & Heflin, J. (2004). An evaluation of knowledge base systems for large OWL datasets. In S.A. McIlraith & D. Plexousakis (Eds.), *3rd International Semantic Web Conference (ISWC'04),* Hiroshima, Japan, (*LNCS* 3298, pp. 274-288). Berlin: Springer-Verlag.

Hermenegildo, M., Bueno, F., Cabeza, D., Carro, M., García, M., López, P., & Puebla, G. (2000). The ciao logic programming environment. In J.W. Lloyd, V. Dahl, U. Furbach, M. Kerber, K. Lau, C. Palamidessi, L.M. Pereira, Y. Sagiv, & P.J. Stuckey (Eds.), *International Conference on Computational Logic (CL'00)*, London, UK, (*LNCS* 1861). Berlin: Springer-Verlag.

Horrocks, I., Fensel, D., Harmelen, F., Decker, S., Erdmann, M., & Klein, M. (2000). OIL in a nutshell. In R. Dieng & O. Corby (Eds.), *12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00),* Juan-Les-Pins, France, (*LNAI* 1937, pp. 1-16). Berlin: Springer-Verlag.

Horrocks, I., & van Harmelen, F. (Eds.) (2001). Reference description of the DAML+OIL (March 2001) ontology markup language (Tech. Rep.). Retrieved October 23, 2006, from http://www.daml.org/2001/03/reference.html

Hustadt, U., Motik, B., & Sattler, U. (2004). Reducing SHIQ descrption logic to disjunctive datalog programs. In M.A. Williams (Ed.), *9th International Conference on the Principles of Knowledge Representation and Reasoning (KRR'04)* (pp. 152-162), Whistler, Canada.

IEEE. (1996). *IEEE standard for developing software life cycle processes* (Std 1074-1995). New York: IEEE Computer Society.

Joachims, T. (1998). A probabilistic analysis of the Rocchio Algorithm with TFIDF for text categorization. In D.H. Fisher (Ed.), *14th International Conference on Machine Learning (ICML'97),* Nashville, Tennessee, (pp. 143-151). San Francisco: Morgan Kaufmann Publishers.

Kalfoglou, Y., & Robertson, D. (1999a). Use of formal ontologies to support error checking in specifications. In D. Fensel & R. Studer (Eds.), *11th European Workshop on Knowledge Acquisition, Modelling and Management (EKAW'99),* Dagsthul, Germany, (*LNAI* 1621, pp. 207-224). Berlin: Springer-Verlag.

Kalfoglou, Y., & Robertson, D. (1999b). Managing ontological constraints. In V.R. Benjamins, A. Chandrasekaran, A. Gómez-Pérez, N. Guarino, & M. Uschold (Eds.), *Proceedings 18 of CEUR Workshop IJCAI99 Workshop on Ontologies and Problem-Solving Methods (KRR-5),* Stockholm, Sweden, (pp. 5.1-5.13). Amsterdam, The Netherlands. Retrieved October 23, 2006, from http://CEUR-WS.org/Vol-18/

Kalyanpur, A., Parsia, B., & Hendler, J. (2005). A tool for working with Web ontologies. *International Journal of Semantic Web and Information Systems, 1*(1), 36-49.

Karp, P.D., Chaudhri, V., & Thomere, J. (1999). XOL: An XML-based ontology exchange language. Version 0.3 (Tech. Rep.). Retrieved October 23, 2006, from http://www.ai.sri.com/~pkarp/xol/xol.html

Khan, L. & Luo, F. (2002). Ontology construction for information selection. In C.V. Ramamoorthy

(Ed.), *CV 14th IEEE International Conference on Tools with Artificial Intelligence*. (pp. 122-127), Washington, DC.

Kietz, J.U., Maedche, A., & Volz, R. (2000). A method for semi-automatic ontology acquisition from a corporate intranet. In N. Aussenac-Gilles, B. Biébow, & S. Szulman (Eds.), *Proceedings 51 of EKAW'00 Workshop on Ontologies and Texts, CEUR Workshop*, Juan-Les-Pins, France, pp. 4.1-4.14. Amsterdam, The Netherlands. Retrieved October 23, 2006, from http://CEUR-WS.org/Vol-51/

Kifer, M., Lausen, G., & Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the ACM, 42*(4), 741-843.

Klein, M. & Fensel, D. (2001). Ontology versioning on the Semantic Web. In I.F. Cruz, S. Decker, J. Euzenat, & D.L. McGuinness (Eds.), *First International Semantic Web Workshop (SWWS'01)*, Stanford, California.

Klein, M., Fensel, D., Kiryakov, A., & Ognyanov, D. (2002). 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 (EKAW'02)*, Sigüenza, Spain. (*LNAI* 2473, pp. 197-212), . Berlin: Springer-Verlag.

Knublauch, H., Fergerson, R., Noy, N.F., & Musen, M.A. (2004). The protege OWL plugin: An open development environment for Semantic Web applications. In S.A. McIlraith & D. Plexousakis (Eds.), *3rd International Semantic Web Conference (ISWC'04)* (pp. 229-243), Hiroshima, Japan. Lecture Notes in Computer Science 3298. Berlin: Springer-Verlag.

Lassila, O., & Swick, R. (1999). Resource description framework (RDF) model and syntax specification. W3C Recommendation. Retrieved October 23, 2006, from http://www.w3.org/TR/REC-rdf-syntax/

Lenat, D.B., & Guha, R.V. (1990). *Building large knowledge-based systems: Representation and inference in the cyc project*. Boston: Addison-Wesley.

Luke, S., & Heflin, J.D. (2000). *SHOE 1.01. Proposed specification* (Tech. Rep.). University of Maryland, Parallel Understanding Systems Group, Department of Computer Science. Retrieved October 23, 2006, from http://www.cs.umd.edu/projects/plus/SHOE/spec1.01.htm

MacGregor, R. (1991). Inside the LOOM classifier. *SIGART Bulletin, 2*(3), 70-76.

Madhavan, J., Bernstein, P.A., & Rahm, E. (2001). Generis schema matching with Cupid. In P.M.G. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, & R.T. Snodgrass (Eds.), *27th International Conference on Very Large Data Bases* (pp. 49-58), Roma, Italy. San Francisco: Morgan Kaufmann Publishers.

Maedche, A., Motik, B., Stojanovic, L., Studer, R., & Volz, R. (2003). Ontologies for enterprise knowledge management. *IEEE Intelligent Systems, 18*(2), 26–33.

Maedche, A., & Staab, S. (2000). Semi-automatic engineering of ontologies from text. In S.K. Chang & W.R. Obozinski (Eds.), *12th International Conference on Software Engineering and Knowledge Engineering (SEKE'2000)*, Chicago, Illinois.

McGuinness, D., Fikes, R., Rice, J., & Wilder, S. (2000). The chimaera ontology environment. In P. Rosenbloom, H.A. Kautz, B. Porter, R. Dechter, R. Sutton, & V. Mittal (Eds.), *17th National Conference on Artificial Intelligence (AAAI'00)* (pp. 1123-1124), Austin, Texas.

Melnik, S., García-Molina, H., & Rahm, E. (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In D. Georgakopoulos (Ed.), *18th International Conference on Data Engineering ICDE'2002* (pp. 117-128), San José, California.

Mena, E., Kashyap, V., Sheth, A.P., & Illarramendi, A. (1996). OBSERVER: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. In W. Litwin (Ed.), *First IFCIS International Conference on Cooperative Information Systems (CoopIS'96)* (pp. 14-25), Brussels, Belgium.

Mikheev, A., & Finch, A. (1997). A workbench for finding structure in texts. In R. Grishman (Ed.), *5th Applied Natural Language Processing Conference (ANLP'97)*, Washington, DC.

Miller, G.A. (1995). WordNet: A lexical database for English. *Communications of the ACM, 38*(11), 39–41.

Mitra, P., Wiederhold, G., & Kersten. (2000). A graph-oriented model for articulation of ontology interdependencies. In P.C. Lockemann (Eds.), *7th International Conference on Extending Database Technology, EDBT 2000* (pp. 1777-1786).

Morin, E. (1998). Prométhée un outil d'aide a l'acquisition de relations semantiques entre temes. In P. Zweigenbaum (Ed.), *5ème National Conference on Traitement Automatique des Langues Naturelles (TALN'98)* (pp. 172-181), Paris, France.

Morin, E. (1999). Acquisition de patrons lexico-syntaxiques caractéristiques dúne relation sémantique. *TAL (Traitement Automatique des Langues), 40*(1), 143-166.

Motta, E. (1999). *Reusable components for knowledge modelling: Principles and case studies in parametric design*. Amsterdam, The Netherlands: IOS Press.

Neches, R., Fikes, R.E., Finin, T., Gruber, T.R., Senator, T., & Swartout, W.R. (1991). Enabling technology for knowledge sharing. *AI Magazine, 12*(3), 36–56.

Noy, N.F. (2004). Evaluation by ontology consumers. *IEEE Intelligence Systems, 19*(4), 80-81.

Noy, N.F., Fergerson, R.W., & Musen, M.A. (2000). The knowledge model of Protege-2000: Combining interoperability and flexibility. In R. Dieng & O. Corby (Eds.), *12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00),* Juan-Les-Pins, France, (*LNAI* 1937, pp. 17-32). Berlin: Springer-Verlag.

Noy, N.F., & Klein, M. (2002). Ontology evolution: Not the same as schema evolution (Tech. Rep. No. SMI-2002-0926). Stanford, California. Retrieved October 23, 2006, from http://smi-web.stanford.edu/pubs/SMI_Abstracts/SMI-2002-0926.html

Noy, N.F., Kunnatur, S., Klein, M., & Musen, M.A. (2004). Tracking changes during ontology evolution. In S.A. McIlraith & D. Plexousakis (Eds.), *3rd International Semantic Web Conference (ISWC'04),* Hiroshima, Japan, (*LNCS* 3298, pp. 259-273). Berlin: Springer-Verlag.

Noy, N.F., & Musen, M.A. (2000). PROMPT: Algorithm and tool for automated ontology merging and alignment. In P. Rosenbloom, H.A. Kautz, B. Porter, R. Dechter, R. Sutton, & V. Mittal (Eds.), *17th National Conference on Artificial Intelligence (AAAI'00)* (pp. 450-455), Austin, Texas.

Noy, N.F. & Musen, M.A. (2001). Anchor-PROMPT: Using non-local context for semantic matching. In A. Gómez-Pérez, M. Grüninger, H. Stuckenschmidt, & M. Uschold (Eds.), *IJCAI'01 Workshop on Ontologies and Information Sharing* (pp. 63-70), Seattle, Washington.

Noy, N.F. & Musen, M.A. (2004a). Specifying ontology views by traversal. In S.A. McIlraith & D. Plexousakis (Eds.), *3rd International Semantic Web Conference (ISWC'04),* Hiroshima, Japan, (*LNCS* 3298, pp. 713-725). Berlin: Springer-Verlag.

Noy, N.F. & Musen, M.A. (2004b). Ontology versioning in an ontology-management framework. *IEEE Intelligent Systems, 19*(4), 6-13.

ODM. (2005). *Ontology definition metamodel*. Third Revised Submission to OMG/ RFP ad/2003-

03-40. Retrieved October 23, 2006, from http://www.omg.org/docs/ad/05-08-01.pdf

Pan, R., Ding, Z., Yu, Y., & Peng, Y. (2005). A bayesian network approach to ontology mapping. In *Proceedings of the 4th International Semantic Web Conference (ISWC'05),* Galway, Ireland, (*LNCS* 3729, pp. 563-577). Berlin: Springer-Verlag.

Raggett, D., Le Hors, A., & Jacobs, I. (1999). HTML 4.01 specification. W3C Recommendation. Retrieved October 23, 2006, from http://www.w3.org/TR/html401/

Rumbaugh, J., Jacobson, I., & Booch, G. (1998). *The unified modeling language reference manual*. Boston: Addison-Wesley.

Schreiber, A.Th., Wielinga, B.J., & Jansweijer, W. (1995). The KACTUS view on the "O" world. In D. Skuce (Ed.), *IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing* (pp. 15.1–15.10).

Shvaiko, P., Giunchiglia, F., & Yatskevich, M. (2004). S-Match: An algorithm and an implementation of semantic matching. In D. Fensel & R. Studer (Eds.), *1st European Semantic Web Symposium (ESWS'04),* Heraklion, Greece, (*LNCS* 3053, pp. 61-75). Berlin: Springer-Verlag.

Staab, S., Schnurr, H.P., Studer, R., & Sure, Y. (2001). Knowledge processes and ontologies. *IEEE Intelligent Systems, 16*(1), 26–34.

Steve, G., Gangemi, A., & Pisanelli, D.M. (1998). Integrating medical terminologies with ONIONS methodology. In H. Kangassalo & J.P. Charrel (Eds.), *Information Modeling and Knowledge Bases VIII*. Amsterdam, The Netherlands: IOS Press. Retrieved October 23, 2006, from http://ontology.ip.rm.cnr.it/Papers/onions97.pdf

Stojanovic, L. (2004). *Methods and tools for ontology evolution*. Doctoral Thesis, FZI, Karlsrhue, Germany.

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

Stumme, G., & Maedche, A. (2001). FCA-MERGE: Bottom-up merging of ontologies. In B. Nebel (Ed.), *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001),* Seattle, Washington, (pp. 225-234). San Francisco: Morgan Kaufmann Publishers.

Sure, Y., Erdmann, M., Angele, J., Staab, S., Studer, R., & Wenke, D. (2002). OntoEdit: Collaborative ontology engineering for the Semantic Web. In I. Horrocks & J.A. Hendler (Eds.), *First International Semantic Web Conference (ISWC'02),* Sardinia, Italy, (*LNCS* 2342, pp. 221-235). Berlin: Springer-Verlag.

Swartout, B., Ramesh, P., Knight, K., & Russ, T. (1997). Toward distributed use of large-scale ontologies. In A. Farquhar, M. Gruninger, A. Gómez-Pérez, M. Uschold, & P. van der Vet (Eds.), *AAAI'97 Spring Symposium on Ontological Engineering* (pp. 138-148). Stanford University.

Uschold, M. (1996). Building ontologies: Towards a unified methodology. In I. Watson (Ed.), *16th Annual Conference of the British Computer Society Specialist Group on Expert Systems*, Cambridge, UK. Retrieved October 23, 2006, from http://citeseer.nj.nec.com/uschold96building.html

Uschold, M., & Grüninger, M. (1996). Ontologies: Principles, methods and applications. *Knowledge Engineering Review, 11*(2), 93–155.

Uschold, M., & King, M. (1995). Towards a methodology for building ontologies. In D. Skuce (Ed.), *IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing* (pp. 6.1-6.10), Montreal, Canada.

Welty, C., & Guarino, N. (2001). Supporting ontological analysis of taxonomic relationships. *Data and Knowledge Engineering, 39*(1), 51–74.

Wu, S.H., & Hsu, W.L. (2002). SOAT: A semi-automatic domain ontology acquisition tool from chinese corpus. In W. Lenders (Ed.), *19th International Conference on Computational Linguistics (COLING'02)*, Taipei, Taiwan.

ENDNOTES

1 Component names depend on the formalism. For example, classes are also known as concepts, entities and sets; relations are also known as roles and properties; and so forth.

2 http://www.omg.org/

3 In (28) *specification* is considered as a pre-development activity. However, following more strictly the IEEE standard for software development, the specification activity was considered part of the proper development process. In fact, the result of this activity is an ontology description (usually in natural language) that will be transformed into a conceptual model by the *conceptualization* activity.

4 http://www.ontologymatching.org/

5 http://swoogle.umbc.edu/

6 http://oyster.ontoware.org/

7 http://www.daml.org/ontologies/

8 http://protege.stanford.edu/

9 Other tools (Ontolingua Server, OntoSaurus, WebOnto, etc.) are described in (Gómez-Pérez et al., 2003).

10 http://www-2-cs-cmu.edu/~mccallum/bow/rainbow

11 http://www.lbreyer.com/

12 http://co4.inrialpes.fr/align

13 http://www.wsmo.org/