The Knowledge Engineering Review, Vol. 17:2, 129–156. © 2002, Cambridge University Press
DOI: 10.1017/S0269888902000462 Printed in the United Kingdom

Overview and analysis of methodologies for building ontologies*

MARIANO FERNÁNDEZ-LÓPEZ and
ASUNCIÓN GÓMEZ-PÉREZ

Laboratorio de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo sn., Boadilla del Monte, 28660. Madrid, Spain. Email: mfernandez@fi.upm.es; asun@fi.upm.es

Abstract

The use of methodologies in software and knowledge engineering is very extensive due to their important advantages. In the case of the development of ontologies, until now, several methodological proposals have been presented for building ontologies. Some of these methodologies are designed for building ontologies from scratch or reusing other ontologies without modifying them, concretely, the following cases can be mentioned: the Cyc methodology, the approach proposed by Uschold and King, Grüninger and Fox's methodology, the KACTUS methodology, METHONTOLOGY and the SENSUS methodology. There is even a proposal for re-engineering ontologies, and several proposals for collaborative construction of ontologies.

In this article, we describe the methodologies and check their degree of maturity, contrasting them with respect to the IEEE standard for software development. Before this, we justify to what extent this standard can be used. A conclusion to this study is that there is no completely mature methodological proposal for building ontologies, since there are some important activities and techniques that are missing in all these methodologies. However, all the methodologies do not have the same degree of maturity. In fact, METHONTOLOGY is a very mature methodology. The other conclusion of this article is that, although work to unify proposals can be interesting, maybe several approaches should coexist.

1 Introduction

Methodologies are broadly used in software engineering and knowledge engineering (Downs *et al*., 1998; Gómez-Pérez *et al*., 1997; MAP, 1990; Waterman, 1986; Wielinga, 1992). A methodology is a "comprehensive, integrated series of techniques or methods creating a general systems theory of how a class of thought-intensive work ought be performed" (IEEE, 1995). A method is an "orderly" process or procedure used in the engineering of a product or performing a service (IEEE, 1990). A technique is a "technical and managerial procedure used to achieve a given objective" (IEEE, 1995). Hence the main difference between methods and techniques is that methods require an order and techniques do not. Both methods and techniques are parts of methodologies. The use of methodologies provides a

* This work is partially supported by the European project "OntoWeb Ontology-based information exchange for knowledge management and electronic commerce", reference IST-2001-29243; the European project "MKBEEM (Multilingual Knowledge-Based European Electronic Marketplace)", reference IST-1999-10589; and the CICYT project "Metodología para la Gestión de Conocimiento", reference TIC-980741.

We are grateful to Oscar Corcho and Maria Dolores Rojas for their input concerning several of the methodologies presented in this paper. We are also grateful of the reviewers of this magazine because of their interesting suggestions.

repeatable process, therefore the final success of the project is less sensitive to the previous experience of the development team in similar projects.

However, in the ontological engineering field, the ontology-building process is a craft rather than an engineering activity. Each development team usually follows its own set of principles, design criteria and phases in the ontology development process. Quite a number of research groups are building ontologies, but it is less clear what design decisions and activities are taken and how they contribute to the success (or failure) of the ontologies developed. The absence of agreed guidelines and methods hinders the development of shared and consensual ontologies within and between teams, the extension of a given ontology by others and its reuse in other ontologies and final applications.

Basically, a number of approaches have been reported for developing ontologies. Already 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 modelling), the first methodological outlines were proposed and later refined in Uschold and Grüninger (1996) and Uschold (1996). At the 12th European Conference for Artificial Intelligence (ECAI'96), Bernaras *et al.* (1996) presented a method used to build an ontology in the domain of electrical networks as part of the Esprit KACTUS (KACTUS, 1996) project. METHONTOLOGY (Gómez-Pérez *et al.*, 1996) appeared at the same time and was extended in later papers (Fernández *et al.*, 1997; Gómez-Pérez, 1998; and Fernández-López *et al.*, 1999). In 1997, a methodology was proposed for building ontologies based on the SENSUS ontology (Swartout *et al.*, 1997). Finally, as an extension to METHONTOLOGY, a re-engineering method was proposed (Gómez-Pérez & Rojas, 1999; Fernández-López *et al.*, 2000). All these methodologies do not consider collaborative and distributed construction of ontologies. The first methodology that includes a proposal for collaborative construction was CO4 (Euzenat, 1995; 1996). CO4 provides a protocol for agreeing new pieces of knowledge against the rest of the knowledge architecture, which has been previously agreed. The Knowledge Annotation of the Knowledge Acquisition Community, also known as $(KA)^2$ (Benjamins *et al.*, 1999) promoted the construction of ontologies in a collaborative and distributed way.

Guarino and his colleagues' methodology (Guarino & Welty, 2000; 2001) and Welty & Guarino, 2001) is out of the scope of this paper because this methodology establishes how to analyse and how to clean the taxonomy of a ready-built ontology using a set of principles. However, this methodology does not propose how to build a complete ontology.

Section 2 will present methodologies for building ontologies from scratch or reusing other ontologies without transforming them, Section 3 will present the re-engineering methodology and Section 4 will present methodologies for collaborative construction. Section 5 will then present the analysis of the different ontologies and Section 6 will show the synthesis derived from the former analysis.

2 Methodologies for building ontologies from scratch or reusing other ontologies without transforming them

This section presents, in chronological order, the best-known approaches for both building ontologies from scratch and reusing ontologies as they are in the ontology libraries. The following approaches for ontology construction do not consider collaborative and distributed aspects.

First, we will generally describe the main steps of the development of Cyc (Section 2.1). This will be followed by the description of Uschold and King's proposal (Section 2.2) based on the development of the enterprise ontology. After that, Section 2.3 will gather the formal approach presented by Grüninger and Fox in the development of the TOVE ontology, also in the enterprise modelling domain. Section 2.4 presents the process followed in order to build ontologies on the ESPRIT 8145 KACTUS project. METHONTOLOGY, which has been adopted by FIPA for ontology construction, will be presented in Section 2.5. Finally, an approach for building specific ontologies and knowledge bases from huge ontologies is presented in Section 2.6.

2.1 Cyc

The Cyc methodology arose from experience of the development of the Cyc Knowledge Base (KB), which contains a huge amount of common-sense knowledge, and which is being built upon a core of over a million hand-entered assertions designed to capture a large portion of what people normally consider consensus knowledge about the world.[1] To codify such a KB, the used language is CycL, an augmentation of first-order predicate calculus, with extensions to handle equality, default reasoning, skolemisation, and some second-order features (for example, quantification over predicates). The reason why the Cyc KB can be considered an ontology is because it can be useful as substrate for building different intelligent systems and also as a base for their communication and interoperatibility. The phases for building the Cyc ontology are (Lenat *et al.*, 1990) given below (see also Figure 1):

- The first phase proposes manually coding the explicit and implicit knowledge appearing in the knowledge sources without the help of natural language and learning systems. This first phase is carried out by hand, since current natural language systems and learning machines do not handle enough common-sense knowledge to search for new common-sense knowledge.
- The second phase proposes knowledge codification that is aided by tools using knowledge already stored in the Cyc KB.
- The third phase delegates the majority of the work to the tools. People will only recommend knowledge sources to read, and they will explain the most difficult parts of the text.

Two tasks are carried out in each phase.

1. *Development of a knowledge representation and top-level ontology* containing the most abstract concepts. Terms like *attribute* or *attribute value* are examples of knowledge representation terms, and *thing*, *intangible* or *collection* are other kinds of abstract concept.
2. Representation of the rest of the knowledge using these primitives.

Cyc has been used during experimentation in the High Performance Knowledge Bases (HPKB), a research program to advance the technology of how computers acquire, represent and manipulate knowledge.[2]

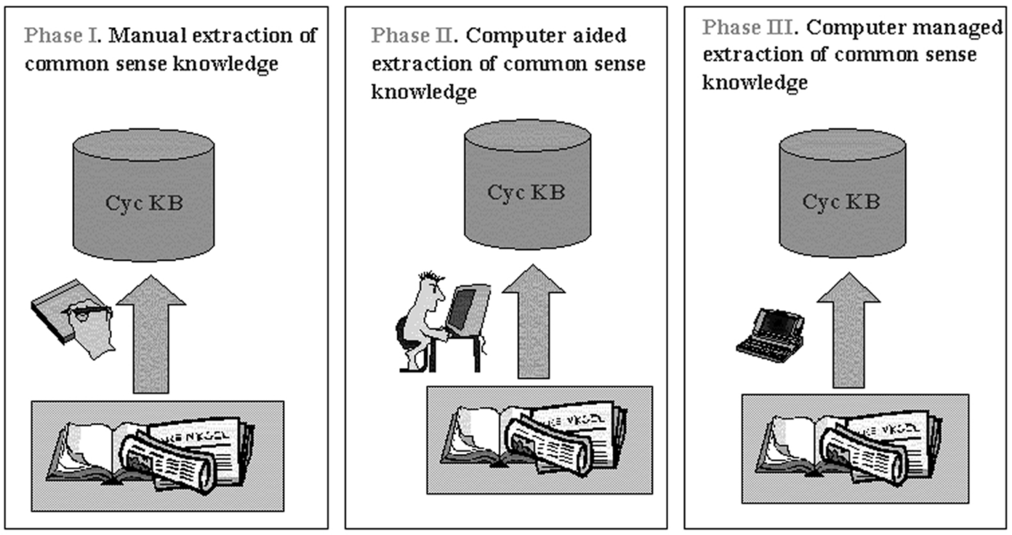
Phase I. Manual extraction of common sense knowledge
Cyc KB
Phase II. Computer aided extraction of common sense knowledge
Cyc KB
Phase III. Computer managed extraction of common sense knowledge
Cyc KB

Figure 1 Phases for building the Cyc knowledge base

[1] `http://www.cyc.com.`

[2] `http://reliant.teknowledge.com/HPKB/about/about.html.`

In how many domains has the Cyc methodology been used? Until now, this methodology is only used for building the Cyc KB; however, Cyc has different micro-theories showing the knowledge of different domains from different viewpoints. In some areas, several micro-theories can be used, and each micro-theory can be seen from different perspectives and with different assumptions.

Concerning the applications in which Cyc ontologies are used, there are several modules integrated into the Cyc KB and the inference engine. One of these modules is the *system of integration of heterogeneous databases*. This module maps the Cyc vocabulary into the schemas of the databases. That is, the data that are stored on the databases are interpreted according to the Cyc vocabulary. The *knowledge-enhanced searching of captioned information* module permits making queries about images using their captions in natural language. Furthermore, the *Guided Integration of Structured Terminology* (GIST) module allows users to import and simultaneously manage and integrate multiple thesauri.

Cyc agents have also been built. All of these agents have a common core with knowledge of Cyc KB, plus particular knowledge in the specific domain of the agent.

Finally, the *WWW information retrieval* module, using the natural language tools that also access the Cyc KB, allows extending the Cyc KB using information available on the Web.

2.2 Methodology of Uschold and King

This methodology is based on the experience of developing the Enterprise Ontology, an ontology for enterprise modelling processes (Uschold & King, 1995) at the Artificial Intelligence Applications Institute (AIAI) of Edinburgh. Although Uschold and King were not the first authors to propose guidelines for building ontologies, they were the first to feel the necessity of using methodologies for ontology development. The steps proposed by the methodology are given below (see also Figure 2).

1. *Identify purpose*. The goal here is to clarify why the ontology is being built and what its intended uses are.
2. *Building the ontology*, which is broken down into three steps:
   2.1. *Ontology capture*. The following tasks are proposed: to identify key concepts and relationships in the domain of interest, to produce precise and unambiguous text definitions for such

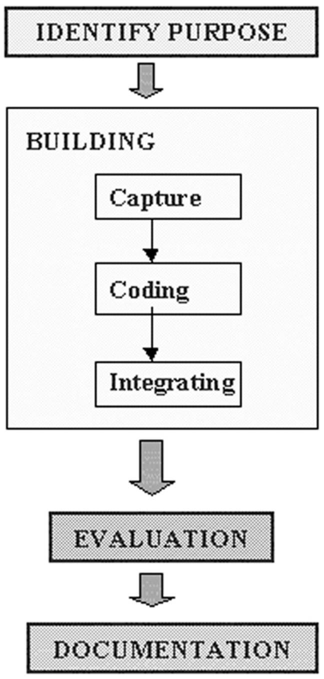

Figure 2 Main steps of Uschold and King's methodology

concepts and relationships and to identify the terms to refer to such concepts and relationships. The authors also identify three possible strategies for identifying concepts: from the most concrete to the most abstract (bottom-up), from the most abstract to the most concrete (top-down), or from the most relevant to the most abstract and most concrete (middle-out).

2.2. *Coding*. This involves explicitly representing the knowledge acquired in step 2.1 in a formal ontology specification language.

2.3. *Integrating existing ontologies*. During either or both the capture and the coding processes, there is the question of how and whether to use ontologies that already exist.

If you follow Uschold and King's approach, one of the main drawbacks during phase 2 (*building the ontology*) is that the developer switches from domain knowledge acquisition into ontology implementation without performing any kind of ontology modelling activities at the knowledge level. The lack of a conceptual model also provokes collateral problems:

P.1) Domain experts and human users have many difficulties understanding ontologies implemented in ontology languages.

P.2) Domain experts are not able to implement ontologies in their domain of expertise. Hence, the knowledge acquisition bottleneck persists.

P.3) Ontologists are forced to code directly into a concrete implementation language. This is an important problem when complex ontologies are being built because ontologists do not have the chance to do an analysis of knowledge that is independent of the technological details needed to model such knowledge. Therefore modelling activities are strongly dependent on the language used to implement the ontology, rather than being independent of the technological details.

3. *Evaluation*, where the authors take the definition from Gómez-Pérez *et al*. (1995, p. 292) and state, "to make a technical judgement of the ontologies, their associated software environment, and documentation with respect to a frame of reference .... The frame of reference may be requirements specifications, competency questions, and/or the real world".
4. *Documentation* recommends that guidelines be established for documenting ontologies, possibly differing according to the type and purpose of the ontology.

One year later, Uschold and Grüninger (1996, pp. 112–113, original emphasis) went deeply into the strategies for identifying concepts on the ontology, and they declared that:

> a bottom-up approach results in a very high *level of detail*. This in turn (1) increases overall *effort*, (2) makes it difficult to spot *commonality* between related concepts, and (3) increases risk of *inconsistencies* which leads in turn to (4) *re-work* and yet more effort.
>
> A top-down approach results in better control of the level of detail, however starting at the top can result in choosing and imposing arbitrary high level categories. Because these are not naturally arising, there is a risk of less *stability* in the model which in turn leads to re-work and greater effort. The emphasis on dividing up rather than putting together also results, for a different reason, in missing the commonality inherent in the complex web of inter-connected concepts.
>
> A middle-out approach, by contrast, strikes a balance in terms of the level detail. Detail only arises as necessary, by specialising the basic concepts, so some effort is avoided. By starting with most important concepts with the most important concepts first, and defining higher level concepts in terms of these, the higher level categories naturally arise and thus are more likely to be stable. This, in turn, leads to less re-work and less overall effort.

According to Uschold and Grüninger, the aforementioned phases are not enough to have a methodology. Every methodology should also include a set of techniques, methods and principles for each of the above four stages, and they should indicate what relationships exist between the stages (e.g. recommended order, interleaving, inputs/outputs).

One of the most important projects developed using this methodology is the Enterprise Ontology, which is a collection of terms and definitions relevant to business enterprises. The ontology was developed under the Enterprise Project by the AIAI at the University of Edinburgh with its partners IBM, Lloyd's Register, Logica UK Limited and Unilever.[3] Besides, a group belonging to Universidad Politécnica de Cataluña, in collaboration with Edinburgh University, has developed WaWO (Waste Water Ontology) in the domain of waste water treatment (Ceccaroni *et al.*, 2000).

The most important tool developed using the Enterprise Ontology is the *Enterprise Toolset*, which uses an agent-based architecture to integrate off-the-shelf tools in a plug-and-play style. The components of the Enterprise Toolset are a *Procedure Builder* for capturing process models, an *Agent Toolkit* for supporting the development of agents, a *Task Manager* for integration, visualisation and support for process enactment, and an Enterprise Ontology for communication.[4]

2.3 Methodology of Grüninger and Fox

This methodology is based on the experience in the development of the TOVE project ontology (Grüninger & Fox, 1995) within the domain of business processes and activities modelling. Essentially, it involves building a logical model of the knowledge that is to be specified by means of the ontology. This model is not directly constructed. First, an informal description is made of the specifications to be met by the ontology and then this informal description is formalised. The steps proposed are as follows (see also Figure 3).

1. *Capture of motivating scenarios*. According to Grüninger and Fox, the development of ontologies is motivated by scenarios that arise in the application. The motivating scenarios are story problems or examples which are not adequately addressed by existing ontologies. A motivating scenario also provides a set of intuitively possible solutions to the scenario problems. These solutions provide an informal intended semantics for the objects and relations that will later be included in the ontology.

   Any proposal for a new ontology or extension to an ontology should describe one or more motivating scenarios, and the set of intended solutions of problems presented in the scenarios.
2. *Formulation of informal competency questions*. These are based on the scenarios obtained in the preceding step and can be considered as expressiveness requirements that are in form of questions. An ontology must be able to represent these questions using its terminology, and be able to

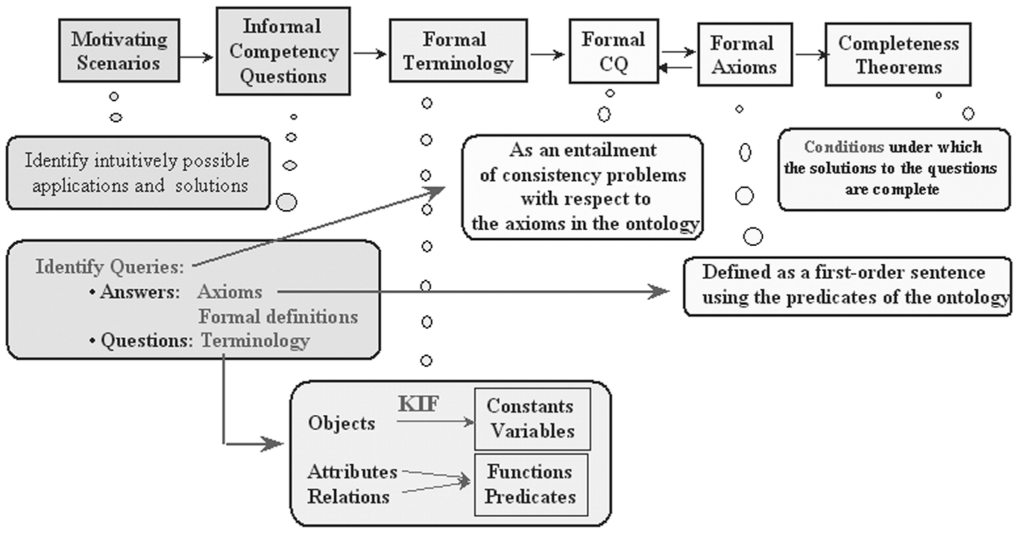
Motivating Scenarios
Informal Competency Questions
Formal Terminology
Formal CQ
Formal Axioms
Completeness Theorems
Identify intuitively possible applications and solutions
As an entailment of consistency problems with respect to the axioms in the ontology
Conditions under which the solutions to the questions are complete
Identify Queries:
• Answers: Axioms
Formal definitions
• Questions: Terminology
Defined as a first-order sentence using the predicates of the ontology
Objects
KIF
Constants
Variables
Attributes
Relations
Functions
Predicates

Figure 3 Main steps of Grüninger and Fox's Methodology

[3] http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html.
[4] http://www.aiai.ed.ac.uk/project/enterprise.

characterise the answers to these questions using the axioms and definitions. These are the informal competency questions, since they are not yet expressed in a given formal language of the ontology.

The competency questions are stratified and the response to one question can be used to answer more general questions from the same or another ontology by means of composition and decomposition operations. This is a means of identifying knowledge already represented for reuse and integrating ontologies.

The questions serve as constraints on what the ontology can be, rather than determining a particular design with its corresponding ontological commitments. There is no single ontology associated with a set of competency questions. Instead, the competency questions are used to evaluate the ontological commitments that have been made to see whether the ontology meets the requirements.

3. *Specification of the terminology of the ontology within a formal language*. The following steps were proposed:
   3.1. *Getting informal terminology.* Once the informal competency questions are available, the set of terms used can be extracted from the questions. These terms will serve as a basis for specifying the terminology in a formal language.
   3.2. *Specification of formal terminology.* Once informal competency questions have been posed for the proposed new or extended ontology, the terminology of the ontology is specified using a formal language such as KIF (Genesereth & Fikes, 1992). These terms will allow the definitions and constraints to be later (step 5) expressed by means of axioms.
4. Formulation of formal competency questions using the terminology of the ontology. Once the competency questions have been informally posed and the terminology of the ontology has been defined, the competency questions are formally defined using the same language.
5. *Specification of axioms and definitions for the terms in the ontology within the formal language*. The axioms in the ontology specify the definitions of terms in the ontology and constraints in their interpretation; they are defined as first-order sentences using axioms to define the terms and constraints for objects in the ontology. Simply proposing a set of objects alone, or proposing a set of ground terms in first-order logic, does not constitute an ontology. Axioms must be provided to define the semantics, or meaning, of these terms.

   If the proposed axioms are insufficient to represent the formal competency questions and characterise the solutions to the questions, then additional objects or axioms must be added to the ontology until it is sufficient. This development of axioms for the ontology with respect to the competency questions is therefore an iterative process.
6. *Establish conditions for characterising the completeness of the ontology.* Once the competency questions have been formally stated, we must define the conditions under which the solutions to the questions are complete.

In summary, we can say that it is a logic-based formal method that transforms informal scenarios expressed in natural language into a computable model expressed in logic.

This methodology was used to build the TOVE (TOronto Virtual Enterprise) project ontologies at the University of Toronto Enterprise Integration Laboratory. These ontologies constitute an integrated model formalised using first-order logic. The TOVE ontologies include Enterprise Design Ontology, Project Ontology, Scheduling Ontology and Service Ontology.[5]

The ontologies built according to this methodology have been used to create a knowledge-aided design system for supporting Design-in-the-Large. The objective of the Design-in-the-Large project is to increase engineering quality and reduce engineering time for multi-person engineering projects. TOVE has been used in the Enterprise Design Workbench, which is a design environment that allows the user to explore a variety of enterprise designs. The process of exploration is one of design, analysis and redesign, where the workbench provides a comparative analysis of enterprise design alternatives,

[5] `http://www.eil.utoronto.ca/eil.html`.

and guidance to the designer. Using the ontologies built with the Grüninger and Fox methodology, the integrated Supply Chain Management Project agents have been developed. The goal is to organise the supply chain as a network of cooperating, intelligent agents, each performing one or more supply chain functions, and each coordinating their actions with other agents. The TOVE virtual enterprise provides the unified testbed used by the agents they built for the major supply chain functions: logistics, transport, management and so on. Further information about these applications is available at `http://www.eil.utoronto.ca/eil.html`.

2.4 The KACTUS approach

The work of Bernaras and her colleagues is set within the Esprit KACTUS project (KACTUS, 1996). One of the objectives of the ESPRIT 8145 KACTUS project is to investigate the feasibility of knowledge reuse in complex technical systems and the role of ontologies to support it (Schreiber *et al.*, 1995).

This approach for developing ontologies is conditioned by application development. Thus, every time an application is built, the ontology that represents the knowledge required for the application is refined. This ontology can be developed by reusing others and can also be integrated into ontologies of later applications. Therefore every time an application is developed, the following steps are taken (Bernaras *et al.*, 1996):

1. *Specification of the application*, which provides an application context and a view of the components that the application tries to model.
2. *Preliminary design based on relevant top-level ontological categories*, where the list of terms and tasks developed during the previous phase is used as input for obtaining several views of the global model in accordance with top-level ontological categories. This design process involves searching ontologies developed for other applications, which are then refined and extended for use in the new application (see Figure 4).
3. *Ontology refinement and structuring*, in order to achieve a definitive design following the modularisation and hierarachical organisation principles. Bernaras *et al.* also recommend that the modules are not very dependent, and that they are as coherent as possible, looking to get maximum homogeneity inside each module.

The experience that illustrates this approach is the KACTUS project. The authors present the development of three ontologies as a result of the development of the same number of applications. The purpose of the first application is to diagnose faults in an electrical network. The second concerns scheduling service resumption after a fault in the electrical network. By the time at which the development of the ontology of this application started, the ontology of faults had also been built. Then the developers refined and augmented the domain concepts by looking at the top-level ontological categories and modifying the relevant domain concepts already identified in the diagnosis ontology,

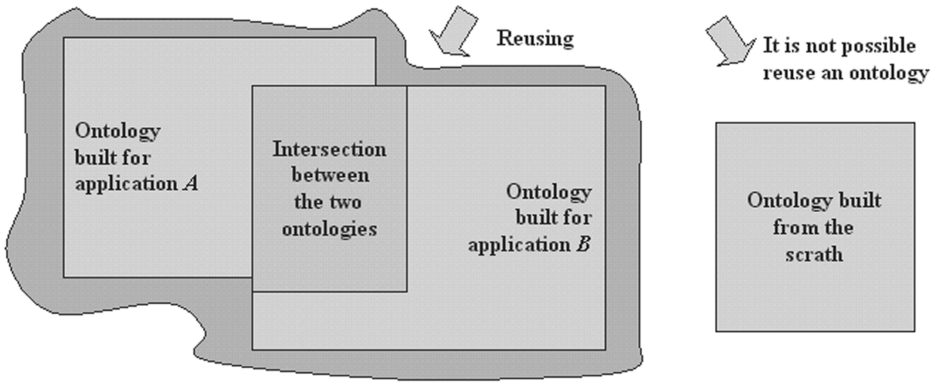
Reusing
It is not possible reuse an ontology
Ontology built for application *A*
Intersection between the two ontologies
Ontology built for application *B*
Ontology built from the scrath

Figure 4 The KACTUS methodology

and to meet the need of restoration. Then, the structure of the ontology was refined and the definitive design was obtained. The third application controls the electrical network. This application is based on the above two applications. The ontology of this application can be considered as the union of the domain ontology for diagnosis and service recovery planning. The union of the ontologies results in a set of sub-ontologies that belong to the intersection and other sets used for one application or the other but not for both at the same time. The sub-ontologies of the intersection have more possibilities of being reused, since the relevant adaptations in these ontologies have had to be made to use them in (at least) two different applications.

2.5 METHONTOLOGY

This methodology was developed within the Laboratory of Artificial Intelligence at the Polytechnic University of Madrid. The METHONTOLOGY framework (Fernández *et al.*, 1997; Fernández-López *et al.*, 1999) enables the construction of ontologies at the knowledge level. METHONTOLOGY has its roots in the main activities identified by the software development process (IEEE, 1996) and knowledge engineering methodologies (Gómez-Pérez *et al.*, 1997; Waterman, 1986). It includes: the identification of the ontology development process, a life cycle based on evolving prototypes (Kendall & Kendall, 1995; 2002), and particular techniques to carry out each activity. The METHONTOLOGY framework is supported by ODE tool (Blázquez *et al.*, 1998; Fernández-López *et al.*, 1999) and WEB-ODE (Arpírez *et al.*, 2001), a scalable workbench for ontological engineering. The first tool is mono-user; the second can work with different users.

The ontology development process (Fernández, 1997) refers to which activities are performed when building ontologies. It is crucial to identify these activities if agreement is to be reached on ontologies that are to be built by geographically distant cooperative teams with some assurance of correctness and completeness. If this is the case, it is advisable to carry out the three categories of activity presented below and steer clear of anarchic constructions (see Figure 5).

Project management activities include scheduling, control and quality assurance. *Scheduling* identifies which tasks are to be performed, how they will be arranged and how much time and what resources are needed for their completion. This activity is essential for ontologies that use ontologies stored in ontology libraries or ontologies that require a high level of abstraction and generality. *Control*

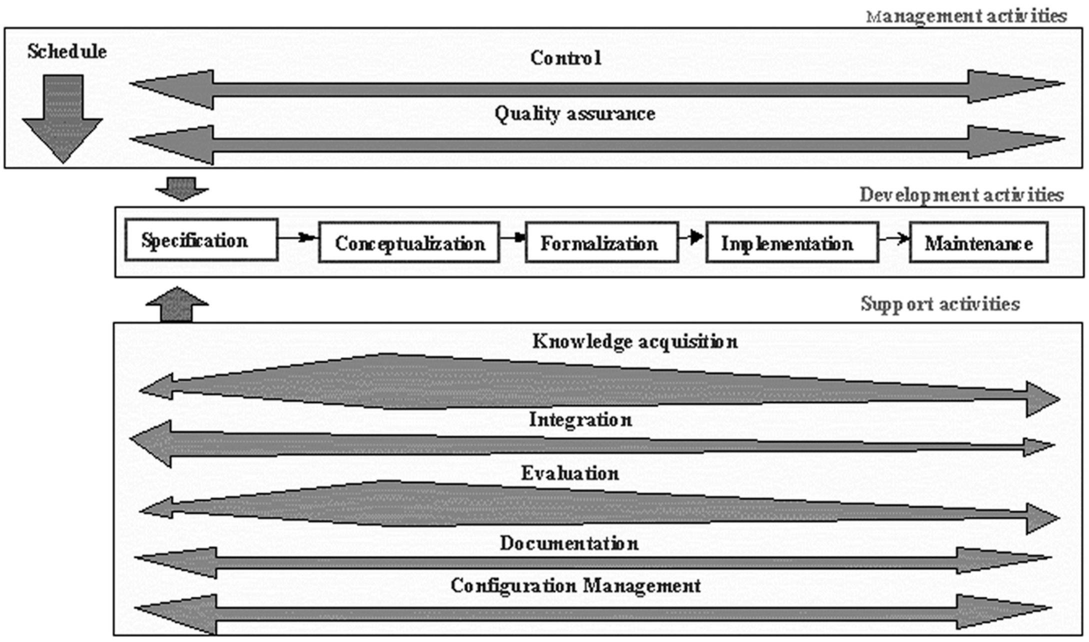

Figure 5 METHONTOLOGY (adapted from Fernández *et al.* (1999))

guarantees that planned tasks are completed in the manner that they were intended to be performed. Finally, *quality assurance* assures that the quality of each and every product output (ontology, software and documentation) is satisfactory.

Development-oriented activities include specification, conceptualisation, formalisation and implementation. *Specification* states why the ontology is being built, what its intended uses are and who the end-users are. *Conceptualisation* structures the domain knowledge as meaningful models at the knowledge level. *Formalisation* transforms the conceptual model into a formal or semi-computable model. *Implementation* builds computable models in a computational language. Finally, *maintenance* updates and corrects the ontology. Detailed descriptions with examples of how most of the development activities are performed appear in Fernández-López *et al.* (1999), Blázquez *et al.* (1998) and Gómez Pérez (1998).

Support activities include a series of activities, performed at the same time as development-oriented activities, without which the ontology could not be built. They include knowledge acquisition, evaluation, integration, documentation and configuration management. *Knowledge acquisition* acquires knowledge of a given domain. *Evaluation* makes a technical judgment of the ontologies, their associated software environments and documentation with respect to a frame of reference during each phase and between phases of their life cycle (Gómez Pérez *et al.*, 1995). *Integration* of ontologies is required when building a new ontology reusing other ontologies that are already available. *Documentation* details, clearly and exhaustively, each and every one of the completed phases and generated products. *Configuration management* records all the versions of the documentation, software and ontology code to control the changes. In Fernández-López *et al.* (1999) and Gómez Pérez *et al.* (1999), descriptions are given of how knowledge acquisition was performed in the CHEMICALS ontology, and evaluation, integration and configuration management are discussed in Gómez-Pérez *et al.* (1999), where the documentation put out is discussed as part of the description of each activity.

On the other hand, the life cycle identifies the *set of stages* through which the ontology moves during its lifetime, describes what activities are to be performed in each stage and how the stages are related (relation of precedence, return, etc.). In Fernández *et al.* (1997), a justification is given of why the ontology life cycle should be based on evolving prototypes. For each prototype, METHONTOLOGY proposes beginning with the *specification* of the ontology. Simultaneously with this phase, the knowledge acquisition activity starts. Once the first prototype has been specified, the construction of the conceptual model is built at the *conceptualisation* phase. It is like assembling a jigsaw puzzle from the pieces supplied by the knowledge acquisition activity, which is completed during the conceptualisation stage. After conceptualisation, *formalisation* and *implementation* of the knowledge are carried out. It is important to say that formalisation is not a mandatory activity, because if you use ODE or WebODE, the conceptualisation model is automatically generated into ontological specification languages.

It can be seen in Figure 5 that control, quality assurance, integration, evaluation, documentation and configuration management are carried out simultaneously into development activities. However, the conceptualisation phase requires most effort in integration and evaluation.

METHONTOLOGY also considers that the different activities to be performed during the development of a specific ontology may involve performing other types of activity on other ontologies already built or under construction (Fernández López *et al.*, 2000) (see Figure 6). For example, the specification of an ontology can provoke modification of another ontology that has to be reused and, therefore, configuration management and documentation of the reused ontology have also to be carried out. This leads to METHONTOLOGY taking into account relationships (or interdependencies) between the life cycle of different ontologies. These interdependencies have as a consequence the unification of ontology development management policies and the integration of products output throughout the development of ontologies whose development processes are interrelated.

METHONTOLOGY has been used at UPM for building the following ontologies:

(A) CHEMICALS (Gómez-Pérez *et al.*, 1996; Fernández-López *et al.*, 1999) is an ontology that contains knowledge within the domain of chemical elements and crystalline structures.

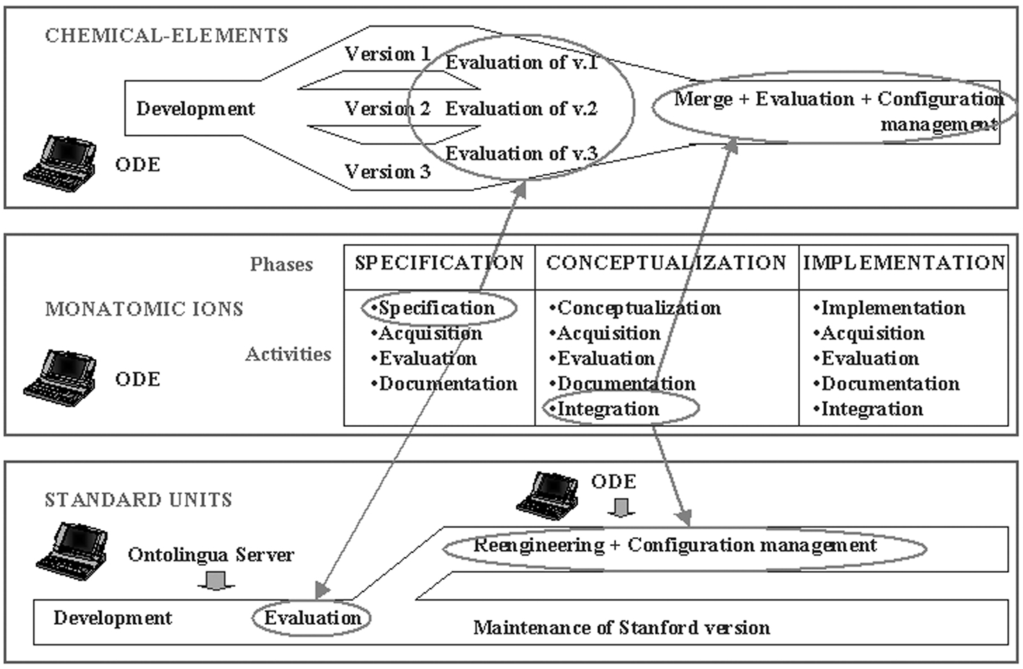

Figure 6 Interdependencies and intradependencies in ontology development

(B) Monatomic ions ontology (Rojas, 1998) gathers information about monatomic ions.

(C) Environmental pollutants ontologies (Mohedano, 2000) represent the methods of detecting the different pollutant components of various media – water, air, soil and so on – and the maximum permitted concentrations of these components, taking into account all the legislation in force (European Union, Spanish, German, US regulations and so on).

(D) The Reference-Ontology (Arpírez *et al.*, 2000) is an ontology in the domain of ontologies that plays the role of a kind of yellow pages of ontologies. It gathers, describes and has links to existing ontologies, using a common logical organisation.

(E) The restructured version of the $(KA)^2$ ontology (Blázquez *et al.*, 1998) contains knowledge about the scientific community in the field of knowledge acquisition, particularly scientists, research topics, projects, universities and so on.

(F) Silicate ontology (Pinilla Ponz, 1999) has a hierarchy of 84 concepts and 17 properties in the domain of the minerals, particularly silicates.

(G) Hardware and software ontology (under construction) is an ontology for knowledge management that provides the necessary vocabulary in the domain of machines, devices and programs inside the Laboratory of Artificial Intelligence.

The following applications were also built: *(Onto)^{2}Agent* (Arpírez *et al.*, 2000) is an ontology-based WWW broker about ontologies that uses the Reference-Ontology as a source of its knowledge and retrieves descriptions of ontologies that satisfy a given set of constraints. *Chemical OntoAgent* (Arpírez *et al.*, 2000) is an ontology-based WWW chemistry teaching broker that allows students to learn chemistry and to test their skills on this domain. It uses CHEMICALS as a source of its knowledge. *Ontogeneration* (Aguado *et al.*, 1998) is a system that uses a domain ontology (CHEMICALS) and a linguistic ontology (GUM) (Bateman *et al.*, 1995) to generate Spanish text descriptions in response to the queries of students in the domain of chemistry. *OntoRoadMap* is a web application that uses the Reference Ontology to allow researchers to consult and update knowledge on ontologies, ontology development methodologies, ontology development tools and ontology applications. OntoRoadMap has been developed inside the European project OntoWeb, a thematic network whose purpose is to bypass communication bottlenecks between different and heterogeneous interest groups that work in ontologies.[6]

This methodology has been proposed for ontology construction by the Foundation for Intelligent Physical Agents (FIPA), which promotes inter-operability across agent-based applications.[7]

2.6 SENSUS-based methodology

This approach is based on the use of a huge ontology (SENSUS) for building specific ontologies and knowledge bases to be used in applications. SENSUS is an ontology to be used in natural language processing. It was developed by the ISI (Information Sciences Institute) natural language group to provide a broad-based conceptual structure for developing machine translators (Knight & Luck, 1994; Knight *et al.*, 1995). Its current content was obtained by extracting and merging information from various electronic sources of knowledge (Swartout *et al.*, 1997). This process (see Figure 7) began by merging manually the PENMAN Upper Model (Bateman, 1994) and ONTOS (Nirenburg & Defrise, 1992) (two, very high-level linguistically based ontologies) and the semantic categories from a dictionary to produce what is called the ontology base. WordNet was then merged (again by hand) with the ontology base. A merging tool was then used to merge WordNet with an English dictionary. Then, to support machine translation, the result of this merger was augmented by Spanish and Japanese lexical entries from the Collins Spanish/English dictionary and the Kenkyusha Japanese/English dictionary.

SENSUS has more than 50,000 concepts organised in a hierarchy according to their level of abstraction. It includes terms with both a high and a medium level of abstraction but, generally speaking, it does not cover terms from specific domains. When a specific domain ontology is being built, the domain terms are pointed out (as seed) in SENSUS in order to build ontologies for particular domains, and any irrelevant terms are pruned in SENSUS (see Figure 8).

Concerning the methodology for building domain ontologies or KB's use of SENSUS, the following steps should be taken (Swartout *et al.*, 1997) (see also Figure 9).

1. A series of terms are taken as *seed*.
2. These seed terms are linked to SENSUS by hand.

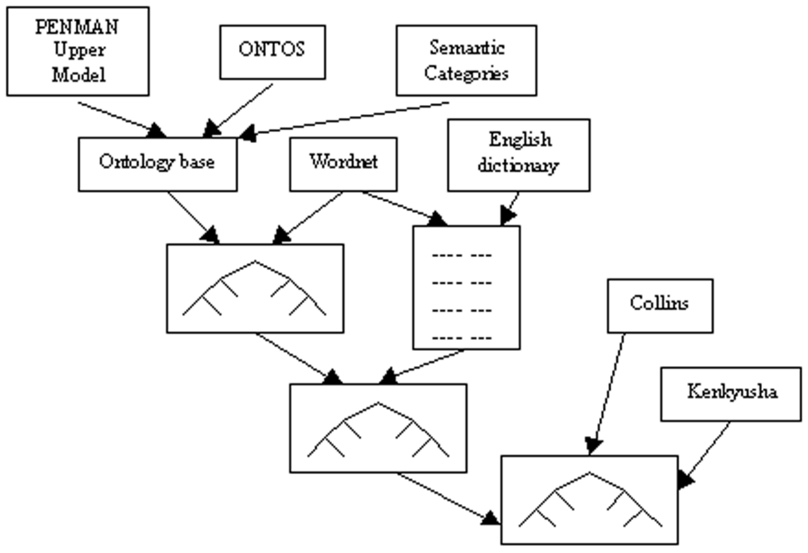

Figure 7 SENSUS ontology building process (Swartout *et al.*, 1997)

[6] http://www.ontoweb.org.
[7] http://www.fipa.org.

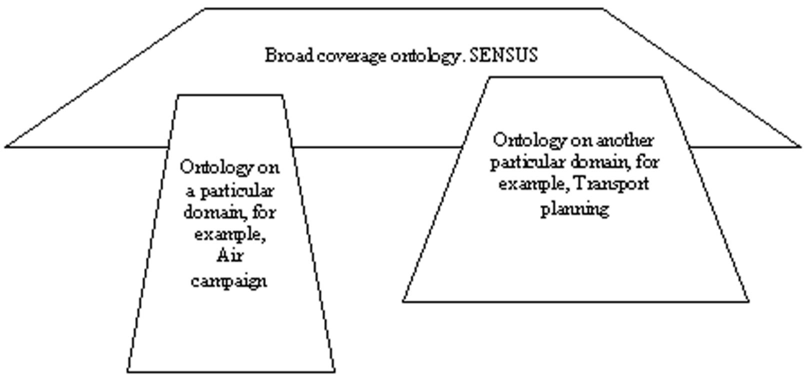
Broad coverage ontology. SENSUS
Ontology on a particular domain, for example, Air campaign
Ontology on another particular domain, for example, Transport planning

Figure 8 Linking terms to SENSUS (adapted from Swartout *et al.* (1999))

3. All the concepts in the path going from the new terms to the root of SENSUS are included.
4. Terms that could be relevant within the domain and have not yet appeared are added.
5. Finally, 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. This step is done manually, since it seems to require some understanding of the domain to make the decision. Obviously, very high-level nodes in the ontology will always have many paths through them, but it is hardly ever appropriate to include the entire subtrees under these nodes.
6. New domain terms are added to complete the base that has been obtained in the previous steps.

This approach promotes shareability of models, because the same base ontology is used to develop ontologies in concrete domains.

An ontology for military air campaign planning has been built using SENSUS. It contains an overview of the basic elements that characterise air campaign plans, such as campaign, scenario, participants, commanders and so on (Valente *et al.*, 1999). It also includes ontologies on weapons, systems in general, fuel and so on.

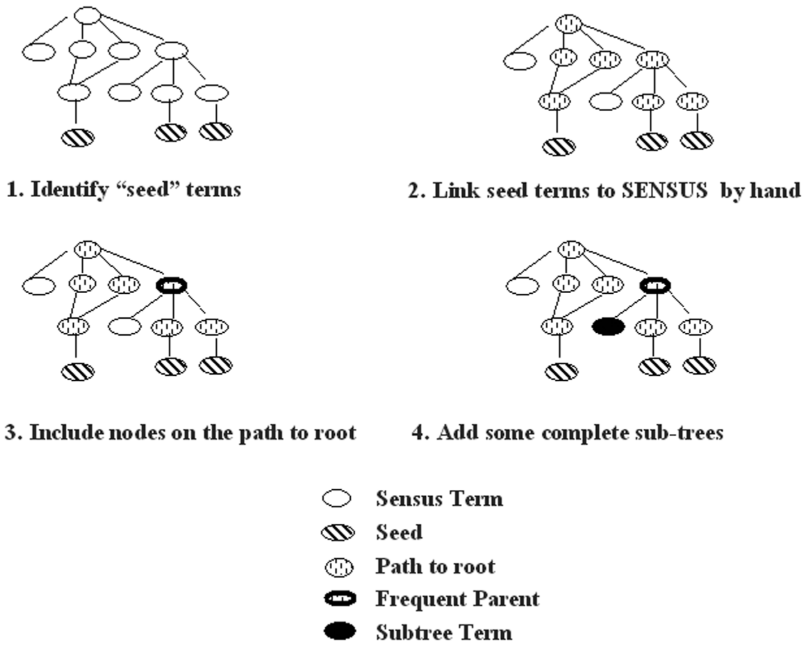
1. Identify "seed" terms
2. Link seed terms to SENSUS by hand
3. Include nodes on the path to root
4. Add some complete sub-trees
Sensus Term
Seed
Path to root
Frequent Parent
Subtree Term

Figure 9 SENSUS methodology (Swartout *et al.*, 1997))

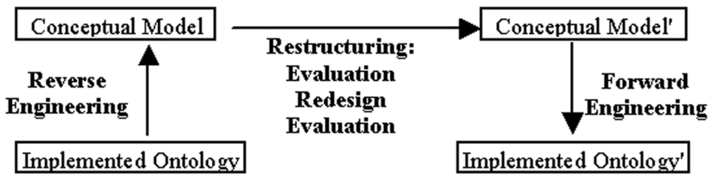

Figure 10 Ontological re-engineering process

The software used for building ontologies following this methodology is Ontosaurus (Swartout *et al.* 1997), which has also been developed at the ISI. It is a Web server accessible through every navigator, and represents knowledge in LOOM (MacGregor, 1992). It permits translation from LOOM into Ontolingua (Farquhar *et al.*, 1997), KIF (Genesereth & Fikes, 1992) and C++.

3 Methodology for re-engineering ontologies

When a given ontology is going to be reused into another ontology (or an application), the conceptual model of the ontology to be reused may not be in accordance with the modelling criteria followed by the group building the new ontology. For example, attributes in the ontology to be reused are global, and in the new ontology might be local. Another important problem is the lack of match between the content of the ontology to be reused and the consensus achieved by the experts in the domain. Even a conflict with used standards can happen. Other times, the conceptual model, or even the design, is missing and a re-engineering process is necessary. However, the most usual problem that forces re-engineering is the lack of documentation explaining the conceptual model of the ontology; documentation explaining the design criteria that were followed to build the ontology and a guarantee of the evaluation process that was followed to check the ontology.

Ontological re-engineering (Gómez-Pérez & Rojas, 1999) is the process of retrieving and mapping a conceptual model of an implemented ontology to another, more suitable conceptual model, which is reimplemented. A proposal of a method for re-engineering ontologies is presented in Figure 10 and adapts Chikofsky's software re-engineering schema (Chikofsky, 1990) to the ontology domain. Three main activities were identified: *reverse engineering*, *restructuring* and *forward engineering*. Figure 11 pictures in detail an organisational chart showing the activities performed during the re-engineering process and the documents generated in each step. A description of each step is presented below.

Reverse Engineering. The objective is to output a possible conceptual model on the basis of the code in which the ontology is implemented. For the purpose of building a conceptual model, the set of intermediate representations proposed by METHONTOLOGY is used. In fact, this re-engineering methodology can be considered an extension of the METHONTOLOGY framework.

Restructuring. The goal of restructuring is to reorganise the initial conceptual model into a new conceptual model which is built bearing in mind the use of the restructured ontology by the ontology or application that reuses it, which means that there is no way of assuring that the restructured ontology will be totally valid for all the ontologies that reuse the restructured knowledge. The restructuring activity contains two phases: analysis and synthesis. The analysis phase includes evaluation (steps 2 to 5 of Figure 11), where the general aim is to evaluate the ontology (Gómez-Pérez *et al.*, 1995), that is, to check that the hierarchy of the ontology and its classes, instances, relations and functions are complete, consistent (there are no contradictions), concise (there are no explicit and implicit redundancies) and syntactically correct. The synthesis phase (step 6 of Figure 11) seeks to correct the ontology after the analysis phase and document any changes made. Hence activities that are related to *configuration management* arise in that context, where the goal is to keep a record of ontology evolution and strict change control.

A series of documents will be generated, which can be divided into three groups: (1) analysis documents, including a list of anomalies (problems, errors, omissions, ambiguities and so on) encountered and detected in steps 2 and 4; (2) configuration management documents, which include

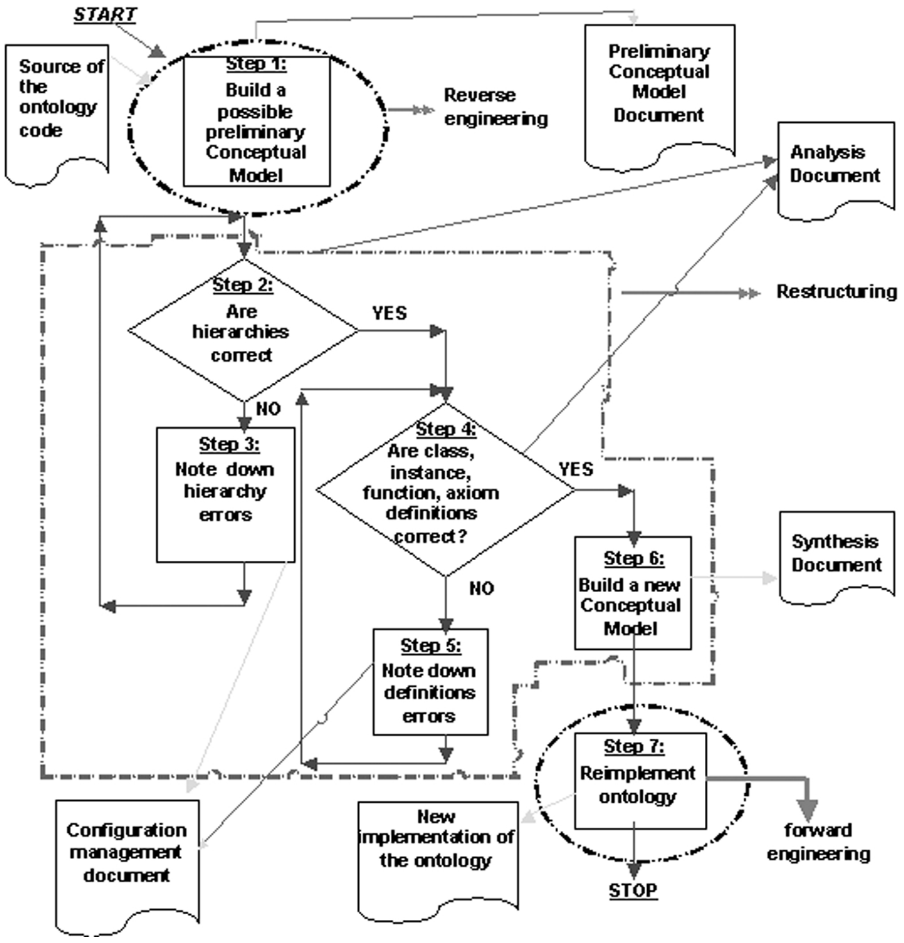
START
Source of the ontology code
Step 1: Build a possible preliminary Conceptual Model
Reverse engineering
Preliminary Conceptual Model Document
Analysis Document
Restructuring
Step 2: Are hierarchies correct
YES
NO
Step 3: Note down hierarchy errors
Step 4: Are class, instance, function, axiom definitions correct?
YES
NO
Step 5: Note down definitions errors
Step 6: Build a new Conceptual Model
Synthesis Document
Step 7: Reimplement ontology
forward engineering
Configuration management document
New implementation of the ontology
STOP

Figure 11 Ontological re-engineering activities

reports related to the changes made in steps 3 and 5 on the basis of the set of errors identified in the analysis documents. These documents include description, need and effects of the change, possible alternatives, justification of the selected alternative, date of the change and so on; and (3) synthesis documents, including the actions taken and criteria observed in step 6.

Forward Engineering. The objective of this step is to output a new implementation of the ontology on the basis of the new conceptual model.

The proposed work method is a sound initial approach to carrying out the above-mentioned process, although it could be improved in later studies using more complex ontologies. In order to increase the reusability of the ontology to be re-engineered, guidelines and criteria to achieve a higher degree of reusability are needed in the restructuring process. Another open issue regards the relationship between the ontology that is being re-engineered and top-level ontologies, if any.

Finally, to keep control of the changes made, *configuration management* should be performed not only during the restructuring process, as we said before, but also during the whole process. For the purpose of assuring information about the evolution of the ontology, a rigorous change control must be performed throughout the restructuring phase. The goal is to have all the changes documented, detailing the changes made and their causes and effects. The configuration management documents can rule out incorrect decision-making, if they state the courses of action to be taken at any time, and justify the choice of one rather than another. Change control also helps end-users to determine which version of the ontology they require for their system or for the new ontology they are to develop.

Change control starts with a petition for change, followed by the classification and registration, approval or initial rejection and evaluation of the change petition, submission of the change report to

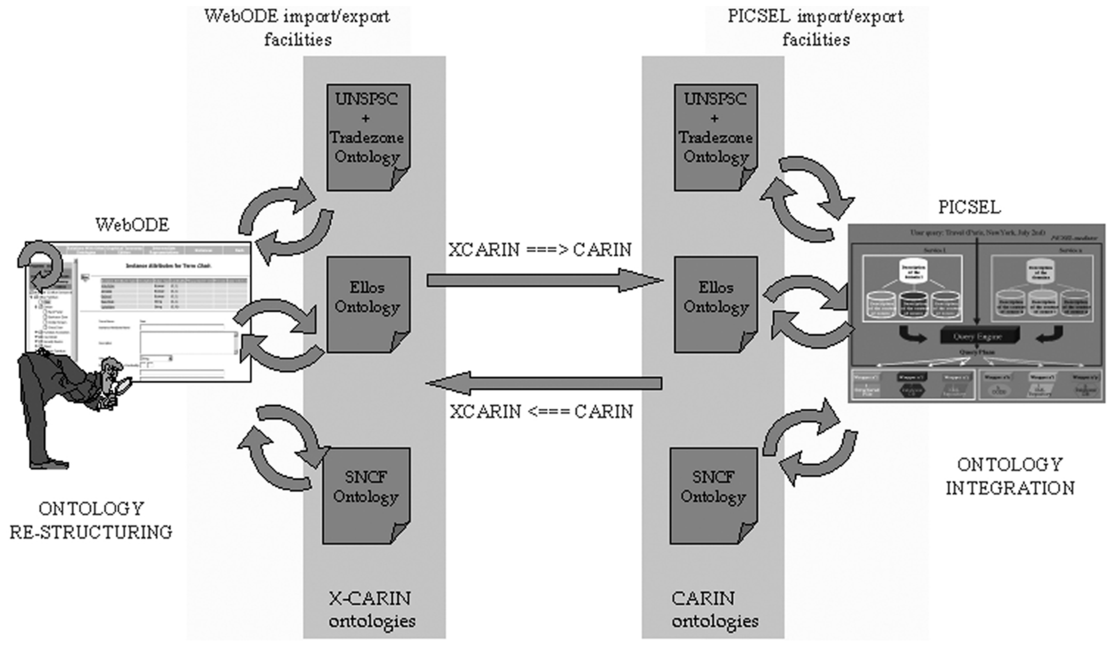

Figure 12 Transformation and integration of different ontologies in the MKBEEM project

the Change Control Committee, performance of the change and certification that the change was made correctly. It ends when the result is reported to the person who proposed the change.

Some of the re-engineered ontologies are Standard Units (Gómez-Pérez & Rojas, 1999) and ontologies that are produced by the IST-1999-10589 MKBEEM project (Leger *et al*., 2000). In the first case, measure units of the International System were needed to build the Monatomic Ions ontology. This fact leads to the reusing of the Standard Units (Gruber & Olsen, 1994) ontology, which is in the Ontolingua Server (Farquhar *et al*., 1997). Given that this ontology was directly implemented in Ontolingua code, it did not have design. Therefore a process of reverse engineering was performed and then, when the experts in chemistry carried out a preliminary examination, and the ontologists also did their revision of the ontology, the processes of restructuring and direct engineering were applied.

In the second case, the MKBEEM ontologies that were developed by different groups and using different languages and tools (CONE (Kankaanpää, 1999) and WebODE (Arpírez *et al*., 2001)) have to be transformed and integrated into another tool (PICSEL (Goasdoué *et al*., 2000)) (see Figure 12). The tasks to be performed are given below.

T.1) Integration of the ontologies in a common knowledge architecture (see Figure 13). This task implies a heavy ontology re-engineering activity, which includes these subtasks: *ontology evaluation*, *ontology integration*, *ontology restructuring*, *building of new translators*, *the elaboration of a method for collaborative construction* and *re-engineering*.

T.2) *Adapting the common knowledge architecture, which has been obtained in task T.1, to the target application*. This adaptation will even imply the extension of X-CARIN (the common interchange format) to make it as expressive as the language of the tool where the ontology will be integrated.

4 Overview of methodologies for cooperative construction of ontologies

An ontology is a shared and common understanding of some domain. Right now, emphasis is put on consensus on the content of ontologies, in the sense that a group of people agrees on the formal specification of the concepts, relations, attributes and axioms that the ontology provides. However, the problems of how to jointly construct an ontology (with a group of people) and how to distributively construct an ontology (with people at different locations) are still unsolved. Euzenat identified the

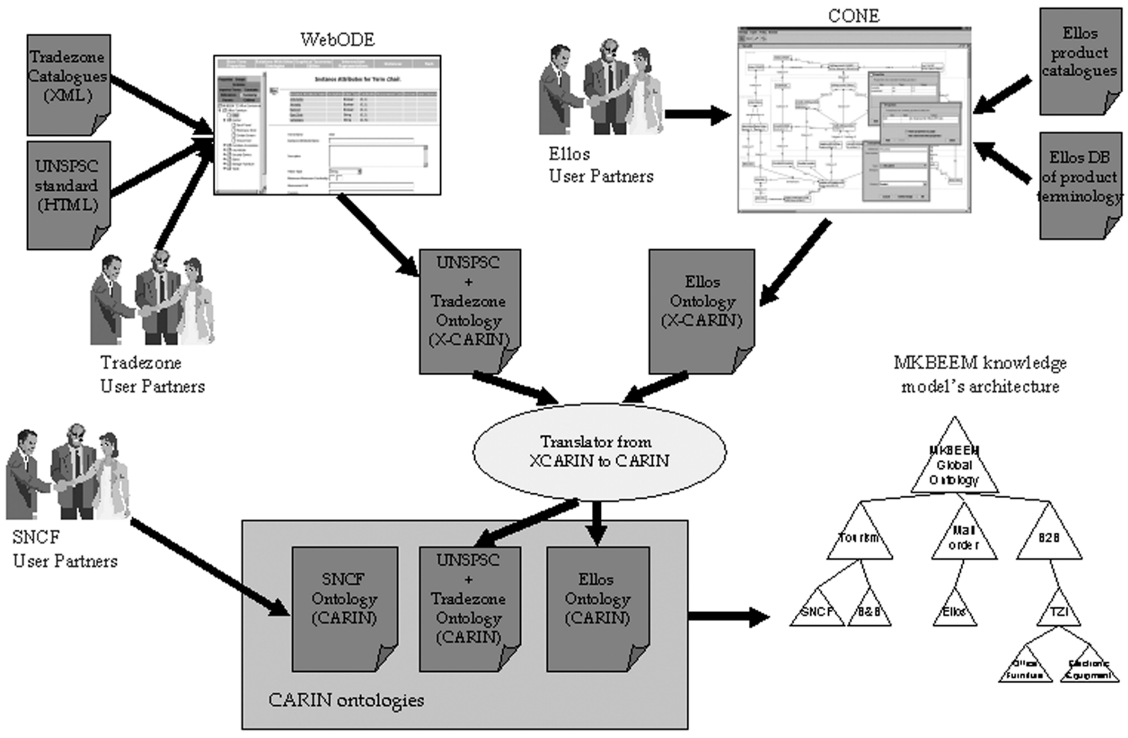

Figure 13 Generation of the common architecture in the MKBEEM project

following problems[8] concerning collaborative construction of ontologies: management of the interaction and communication between people, data access control, recognition of a moral right about the knowledge (attribution), detection and management of errors and concurrent management and modification of the data. The consequence of these problems is that there are few detailed proposals for how to collaboratively build ontologies. Nevertheless, for several years, the main proposals have been CO4, for collaborative construction of KBs at INRIA, and the approach used in ontology building at the Knowledge Annotation Intiative of the Knowledge Acquisition Community, also known as the $(KA)^2$ initiative.

4.1 CO4

CO4 is a protocol to reach consensus between several KBs. Its goal is for people to discuss and agree in the knowledge introduced in the KBs of the system. These KBs are built to be shared, and they have consensual knowledge, hence they can be considered ontologies. Experimentation has been done, above all, in the molecular genetic domain.

According to Euzenat's proposal (Euzenat, 1995; 1996), the KBs are organised in a tree. The leaves are called user KBs, and the intermediate nodes, group KBs. On the one hand, the user KBs do not obligatorily have consensual knowledge. On the other hand, each group KB represents the consensual knowledge among its children (called subscriber KBs). A KB can subscribe to only one group. A human user can create several KBs (possibly subscribing to different group bases) representing different trends, and knowledge can be transferred from one KB to another. Also, it is possible that several human users share the same KB.

When the users of a KB have enough confidence in a piece of knowledge of their KB, and they want to reach consensus about their knowledge with the rest of the users, the following process should be carried out (Figure 14).

[8] `http://www.inrialpes.fr/sherpa/papers/euzenat98c.html`.

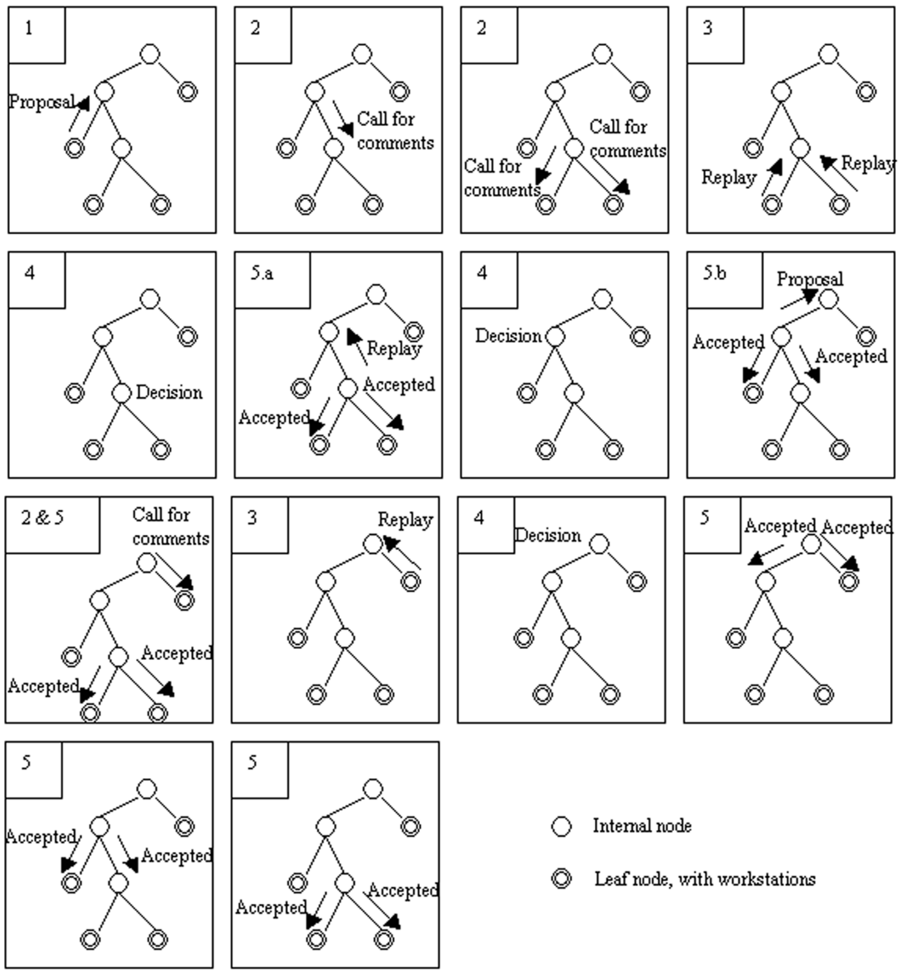

Figure 14 Submitting and acceptance of a user proposal

1. The KB users send a message from his leaf node to the KB group node. The KB group node sends a call for comments to every subscriber node, which is its direct child. A program for evaluating the consistency between the consensual knowledge inside the KB group and the proposed knowledge is used. Another program could also be run to show how to add the proposed knowledge inside the KB group node. This can help users to decide whether or not a given knowledge is accepted.
2. If some of these subscriber nodes are also a KB group, then these nodes send a call for comments to all subscriber nodes that are one of their direct children. This process is repeated recursively for each child node that is also a KB group node, until the leaves of the tree are reached.
3. When the proposal has reached the leaf nodes, each one of them delivers a message to its KB group node accepting or rejecting the proposal, or even making an alternative proposal. To help the users of each leaf node to take the right decision, the program to evaluate the consistency between the knowledge stored in its particular KB and the proposed knowledge can be run.
4. When a KB group receives the replies of all its children, it is decided whether the proposal is accepted or rejected. A possible way to decide is: (a) if every child agrees with the modification, then the proposal is accepted; (b) if the modification is rejected by some children, then it is not accepted in the KB group node (in this case, comments of the users that have rejected the proposal are sent); and (c) if an alternative proposal is delivered from a child, then the initial call for comments is substituted by the call for comments of the later proposal, and it is sent to every child.
5. The subscriber that raised the proposal is notified by the KB group node about the result. If the proposal was accepted, this fact is notified to every group user. Besides, when a decision is made

in a KB group, there are two possible cases: (a) the decision is made because of a call for comments received from the parent. In this case, this call for comments is replayed according to the decision. (b) The decision is made because a child node made a proposal. In this case, if the decision is the proposal acceptance, then such a proposal is sent to the parent as a group reply.

Steps 2, 3, 4 and 5 are repeated until all users accept the proposal, or some user definitively rejects it.

An important aspect of this protocol to reach consensus is that lack of agreement when new knowledge is added does not make it impossible for all users to share it. That is, if a user makes a proposal that does not satisfy the other users, the users and the groups agreeing with the modification can add it to their KBs.

The possibility of subscribing to a new particular KB when the construction of the rest of the KBs has started is also considered. In such a case, the knowledge agreed in its group is introduced in its new particular KB.

Moreover, it is possible for independent KBs to take part as observers. From these nodes, modifications introduced in the tree can be seen, but changes cannot be made from these nodes.

As technological support for this method, the CO4 system (Euzenat, 1996; Euzenat, 1995; Rechenman, 1993) is being built in INRIA. This allows the integration of formal and informal knowledge, and it is accessed via the Web.

4.2 The (KA)^2 methodology

The goal of the Knowledge Annotation Initiative of the Knowledge Acquisition (KA) community, also acknowledged as the (KA)^2 initiative, is to model the knowledge acquisition community using ontologies developed in a joint effort by a group of people at different locations using the same templates and language. The (KA)^2 ontology will form a basis to annotate WWW documents of the knowledge acquisition community in order to enable intelligent access to these documents. (KA)^2 is an open-joint initiative where the participants are actively involved in the distributive ontological engineering development.

The current conceptual model of the (KA)^2 ontology consists of seven related ontologies: an organisation ontology, a project ontology, a person ontology, a publication ontology, an event ontology, a research-topic ontology and a research-product ontology.

In the collaborative and distributed process that was used to build the ontologies, two kinds of people were involved. They were called *ontopic agents* and *ontology coordinating agents*.

Ontopic agents were research groups that have a deep knowledge about some topics of interest. There exist about 15 groups of ontopic agents, each group focusing on a particular topic of the (KA)^2 ontologies. Their aim was to establish a consensual ontology of the KA community, hence they had to have a shared vision of the topic being represented.

The majority of the (KA)^2 ontologies (organisations, projects, people, publications, events, and research-products) were directly developed by the ontology coordinating agents. To build these ontologies, they created their conceptual structure and identified their main concepts, taxonomies, relations, functions, attributes and axioms, delegating the addition of instances to the community. However, the development of the Research-Topic ontology was carried out jointly by ontopic agents, who were experts of the research topic domain and the ontology coordinating agent (see Figure 15). To ease the process of building the Research-Topic ontology, the ontology coordinating agent distributed a template among the ontopic agents, which used e-mail in their intracommunication and also to send their results to the coordinating agents. The ontology was generated from the knowledge introduced via the template. Once the ontology coordinating agents got all the portions of the ontologies from the ontopic agents, they integrated them. In this case, the integration process was not difficult since all the ontopic agents used the same pattern.

A first release of the (KA)^2 ontology was built in FLogic (Kifer *et al*., 1995), which is the ontology language used by Ontobroker (Fensel *et al*., 1998). A decision was made to translate the whole ontology to Ontolingua to make it accessible to the entire community through the Ontolingua server.

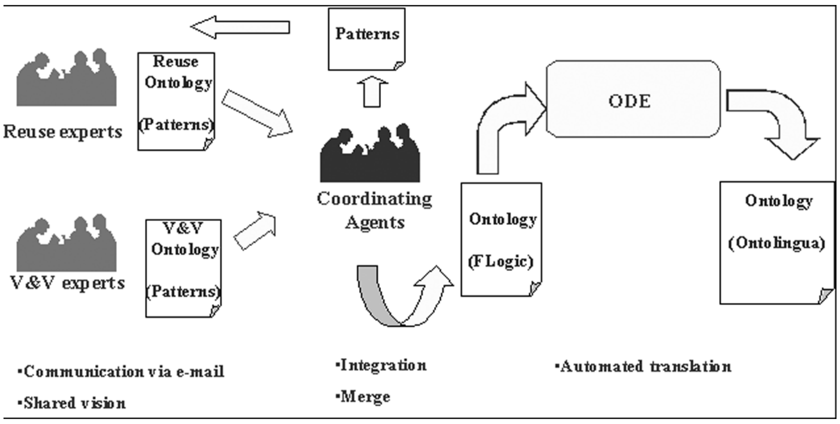

Figure 15 Distributive construction of an ontology in (KA)2

Translation between FLogic and Ontolingua formats of the ontology was performed automatically by translators, which form part of ODE.

5 Analysis of methodologies

Once the main approaches for building ontologies have been presented, in this section we will analyse the coverage and compliance of each methodology with respect to the *IEEE Standard for Developing Software Life Cycle Processes*, 1074-1995 (IEEE, 1996). We also analyse the details of the ontology life cycle.

Previous to the description of the IEEE 1074-1995 standard, we justify why that standard can be applied to the ontology development process. According to the IEEE definition (IEEE, 1990), software is "computer programs, procedures, and possibly associated documentation and data pertaining to the operation of a computer system"; ontologies are part (sometimes only potentially) of software products. Therefore ontologies should be developed according to the standards that are proposed for software, which should be adapted to the special characteristics of ontologies.

Some might say, against the argument of the last paragraph, "ontologies can be used to share knowledge among humans, as pointed out by Uschold and King (1996). Would you also build these ontologies considering a Software Engineering standard?" The answer is yes, of course. According to the Studer *et al*. (1998) definition of ontology, an ontology has to be machine-readable; consequently, although some ontologies can be used by people, they can also be used by software systems.

5.1 IEEE standard 1074-1995

The IEEE 1074-1995 standard describes the software development process, the activities to be carried out and the techniques that can be used for developing software. The activities are not presented in chronological order, since the standard recommends that they be incorporated into a software life cycle, which is selected and established by the user for the project under development. The standard does not define a particular life cycle. These activities are part of what is called the software process, which is further broken down into four main processes. These processes are shown below.

Software life cycle model process. This includes the activities of identifying and selecting a software life cycle, which establishes the order in which the different activities involved in the process should be performed.

Project management processes. These create the framework for the project and ensure the right level of management throughout the entire product life cycle. Activities related to project initiation, project monitoring and control, and software quality management belong to this group of processes.

Software development-oriented processes produce, install, operate and maintain the software and retire it from use. They are divided into the groups given below.

- *Pre-development processes* are performed before starting software development proper. They involve activities that are related to the study of the environment in which the software has to operate, and to feasibility studies.
- *Development processes* are those that must be performed to build the software product. These processes include *requirements process*, which includes iterative activities that are directed towards developing the software requirements specification; *design process*, the goal of which is to develop a coherent and well-organised representation of the software system that meets the requirements specification; and *implementation process*, which transforms the design representation of a software product into a programming language realisation.
- *Post-development processes* are related to the installation, operation, support, maintenance and retirement of a software product. They are performed after software construction.
- *Integral processes* are needed to successfully complete software project activities. They ensure the completion and quality of project functions. They are performed at the same time as software development-oriented processes and include activities that do not output software but are absolutely necessary to obtain a successful system. They cover the processes of verification and validation, software configuration management, documentation development and training.

5.2 How can the IEEE standard be applied to ontology development?

Below, we describe to what extent the IEEE standard processes should be applied in an ontology development methodology.

Software life cycle model process. Proposing activities is not enough for defining a methodology; setting up when each activity is carried out and setting up the relationships among activities is needed in the ontology development process.

Project management processes. Carrying out, or not, the management activities proposed by the standard is not optional. If you build a complex ontology you will need to manage several resources, and you will usually have a limited time for obtaining results.

Software development-oriented processes. Ordered according to process types, there are:

- *Pre-development processes*. Apart from studying the software environment in which the ontology is to be installed, the possibilities of integrating the ontology into other systems also have to be reviewed. The feasibility study is applicable to any software type, although it will vary from one type to another.
- *Development processes*. By process types, it can be said that as shown in Gómez-Pérez (1998), ontologists are able to specify, at least partially, what is expected of the ontology, therefore, a *requirements process* is suitable. As with traditional knowledge bases, direct coding of the result of knowledge acquisition is too abrupt to be performed in a step, especially in the case of complex ontologies. That is, ontologies also have to be designed; consequently a *design process* is also suitable. Obviously, if ontologies are to be used by computers, they have to be implemented, hence an *implementation process* is needed.
- *Post-development processes*. There is not usually an ontology installation, but an ontology integration inside a system. Nevertheless, the ontology operates, needs support, is maintained and is retired simultaneously with the whole system.

Integral processes. An ontology must be evaluated and documented, and configuration management is necessary. Training is also needed for the instruction of people who are responsible for maintaining ontologies, and to improve the knowledge of people who participate in ontology building.

5.3 Compliance of each methodology with the IEEE standard

In this section we compare the compliance of each methodology with respect to the IEEE standard. We summarise this analysis in Table 1. The rows in this table represent the different methodologies, and the columns represent the different processes proposed by the IEEE standard. Each cell of the table can be filled in with three types of value. The value "proposed" means that the methodology of the corresponding row identifies the process that is written in the column as a process to be performed during the ontology development process. The value "not proposed" means that public documentation does not mention the non-considered aspect. If the methodology partially deals with the process, we include the considered activities. According to this table, there is no methodology with a perfect compliance with respect to the IEEE standard. The particular absences in the different groups of processes are given below.

Project management processes. Only METHONTOLOGY and the methodology for re-engineering propose carrying out scheduling, control and quality assurance. However, they do not propose how to establish the project environment.

Project development-oriented processes. The main absences are:

- *Pre-development processes* (identification of the needs, feasibility study and so on). No methodology provides proposals.
- *Development processes* (requirements, design and implementation process). Most of the methodologies do not propose a design process (in particular, Cyc, Uschold and King's methodology, SENSUS, $(KA)^2$ methodology and CO4). Without a process of design, the transition from knowledge acquisition towards codification is too sudden (see Section 2.2).
- *Post-development processes*. Only METHONTOLOGY and the methodology for re-engineering propose carrying out maintenance, although other processes (installation, operation, support, retirement and training) are missing.

Integral processes. Most of the methodologies (except the KACTUS and SENSUS methodologies) establish, at least, the necessity of performing some integral processes. However, all the methodologies have shortfalls in this kind of process.

According to Table 1, METHONTOLOGY and the methodology for re-engineering have much more compliance with the standard than the other methodologies. In fact, they have been built with this standard in mind.

In conclusion we can say that most of the methodologies focus on development activities, especially on the codification of the ontology, and they do not pay attention to other important aspects related to management. This is because ontological engineering is relatively new. However, contributions in management activities and even the design process are needed to improve most of the methodologies. Nevertheless, we also have to say that the shortfalls of the methodologies are due, in most cases, to the purpose of the authors being to give some general guidelines or to explain their experience focusing on the most important aspects. Besides, CO4 and $(KA)^2$ establish protocols for collaborative construction of ontologies, and they can be complemented with other methodologies.

5.4 Life cycle proposals

In this section we comment on the life cycles that are proposed by the methodologies. In software engineering, the main life cycles are the waterfall life cycle defined by Royce (1987), where the activities are *sequential* in the sense that you cannot move on to the next activity until you have completely finished the previous one; the incremental life cycle (McCracken & Jackson, 1982), where software grows by layers, allowing the inclusion of new functions; finally, the evolving prototype life cycle (Kendall & Kendall, 2002), where software grows depending on the needs; indeed, this model

Table 1 Methodology compliance with IEEE 1074–1995 (continuation).

	Project management processes	**Project development-oriented processes**					**Integral processes**
		Pre-development Processes	**Development process**			**Post-development processes**	
			Require-ments process	**Design process**	**Imple-mentation process**		
Cyc	Not proposed	Not proposed	Not proposed	Not proposed	Proposed	Not proposed	Activities not identified for: training, configuration management, and verification and validation
Uschold and King	Not proposed	Not proposed	Proposed	Not proposed	Proposed	Not proposed	Activities not identified for: training, environment study, and configuration management
Grüninger and Fox	Not proposed	Not proposed	Proposed	Proposed	Proposed	Not proposed	Activities not identified for: training, environment study, and configuration management
KACTUS	Not proposed	Not proposed	Proposed	Proposed	Proposed	Not proposed	Not proposed
METHONTOLOGY	Establish project environment activity is not identified	Not proposed	Proposed	Proposed	Proposed	Activities not identified for: installation, operation, support, retirement, and training	Activities not identified for training and environment study.
SENSUS	Not proposed	Not proposed	Proposed	Not proposed	Proposed	Not proposed	Not proposed
Methodology for re-engineering ontologies	Establish project environment activity is not identified	Not proposed	Proposed	Proposed	Proposed	Activities not identified for: installation, operation, support, retirement, and training	Activities not identified for training and environment study.
$(KA)^2$ methodology	Not proposed	Not proposed	Not proposed	Not proposed	Proposed	Not proposed	Not proposed
Co4	Not proposed	Not proposed	Not proposed	Not proposed	Proposed	Not proposed	Only identifies verification

lets you to modify, add and remove functions in the software at any time. The proposals about life cycles that appear (explicitly or implicitly) in the methodologies presented in this paper are given below.

- The phases for building Cyc are described in a general way, and it is assumed that its life cycle is based on evolving prototypes, although this is not explicitly stated in the documentation.
- Uschold and King's methodology does not propose a life cycle for building ontologies.
- Grüninger and Fox's methodology does not propose any life cycle model; however, the order in which the development activities are performed is established, and provision is also made for extending an ontology that has already been built, starting again with getting scenarios. Nevertheless, Grüninger and Fox do not have any statement concerning whether or not the definitions already contained in an ontology built using their methodology can be modified when extending this ontology. Accordingly, it is impossible to ascertain whether the methodology admits development by means of evolving prototypes or only an incremental life cycle.
- Bernaras and her colleagues simply seem to assume evolved prototypes, although they do not explicitly say so.
- METHONTOLOGY clearly proposes an evolving prototypes life cycle.
- SENSUS methodology does not mention any preference for a particular life cycle model. SENSUS is used for automatically building specific domain ontologies, and for these ontologies, nothing is said about how to develop versions other than the first.
- The re-engineering methodology proposes a waterfall life cycle for carrying out the steps of reverse engineering, restructuring and forward engineering. However, the experience of the MKBEEM project is forcing this model to change to become an evolved prototypes model.
- CO4 implicitly proposes evolving prototypes.
- $(KA)^2$ sets up the order in coordination activities, but it does not propose an order for development inside the work groups who build the ontologies. In addition, $(KA)^2$ does not propose how to build new versions of the ontologies.

6 Conclusions

The methodologies presented in this paper can be classified according to the following criteria:

a. *The methodologies that have a proposal for collaborative construction of ontologies*. As we said before, an ontology captures consensual knowledge, that is, it is not private to some individual, but accepted by a group. Although ontologies are built jointly by people geographically separated, a few works, CO4 and $(KA)^2$, related to collaborative construction of ontologies, have been proposed. The rest of the methodologies do not consider collaborative construction in an explicit way.
b. If we compare the methodologies taking into account whether the ontology is used as it is or is substantially modified, on the one hand, the following methodologies *build ontologies from scratch, or reuse other ontologies without transforming them*: Cyc, Uschold and King's methodology, Grüninger and Fox's methodology, KACTUS, METHONTOLOGY and SENSUS. On the other hand, METHONTOLOGY includes a proposal of a methodology for *re-engineering ontologies* (see Section 3), whose goal is to retrieve and transform a conceptual model of an existing and implemented ontology into a new conceptual model which is reimplemented.
c. Finally, taking into account the degree of dependency with respect to the application that uses the ontology, the methodologies can be classified into *application-dependent methodologies*, for example KACTUS, which approaches the ontology construction on the basis of an application knowledge base, by means of a process of abstraction; *application-semidependent methodologies*, for instance, Grüninger and Fox and SENSUS methodologies, which identify, in the specification stage, possible scenarios of ontology use; and *application-independent methodologies*, where the process is totally independent of the uses to which the ontology will be put in knowledge-based

systems, agents and so on. Cyc, Uschold and King's methodology and METHONTOLOGY are examples of the last type of approach.

According to the analysis that has been carried out in the last sections, we arrived at the following conclusions:

1. None of the methodologies are fully mature if we compare them with the IEEE standard; although the following scale can be established:
   i. METHONTOLOGY is the most mature; however, recommendations for the pre-development processes are needed, and some activities and techniques should be specified in more detail. It is recommended by FIPA. Additionally, METHONTOLOGY also includes detailed recommendations for re-engineering ontologies.
   ii. The strength of Grüninger and Fox's methodology is its high degree of formality. However, it does not include guidelines for re-engineering ontologies. Neither is the life cycle specified. Furthermore, although ontologies have been developed with this methodology, and there are also applications that use these ontologies, the domain is confined to business.
   iii. Uschold and King's methodology has the same omissions as the above methodology and is less detailed.
   iv. SENSUS-based methodology, apart from having the shortcomings of the above methodologies, does not mention the life cycle.
   v. The Bernaras *et al.* methodology, apart from having the above omissions, has not been used to build many ontologies and applications.
   vi. CO4 and (KA)^2 provide some guides for building ontologies in collaborative way, but many details do not appear. They propose a politics for the collaboration of different groups, but not for how each particular group works.
2. **The proposals are not unified**. At present each group applies its own methodology. This is exacerbated by the fact that none have reached maturity. Therefore efforts are required along the lines of unifying methodologies to arrive at a situation resembling knowledge and software engineering.

   A preliminary attempt to unify two methodologies was described in Uschold (1996), cited in Section 2. Its disadvantage was that the new synthesised proposal was not an actual methodology, it was a conception of a potential methodology.
3. **There are some approaches completely different from the others**. This may mean that the best we can do is to have several widely accepted methodologies rather than just one standardised methodology.
4. **Each group uses its own methodology, and it is odd to find someone who uses a methodology elaborated by a different group**. That is, if different cross-references from different authors are consulted, it can be checked that other methodologies are not used in their applications.
5. However, **a integration between methodologies for collaborative work and other methodologies would be very interesting**. For instance, METHONTOLOGY people participate in (KA)^2 project, developing ontologies, therefore integrating METHONTOLOGY with (KA)^2 methodology should not be very difficult.
6. **Interoperatibility is allowed between systems**. Domain ontologies built using the SENSUS approach share the same high-level concepts (or skeleton). Hence systems that use such ontologies will share a common structure of the world, and it would be easier for them to communicate because they share the same underlying structure.
7. Except for METHONTOLOGY and the methodology for re-engineering, the rest of the methodologies are not described in a detailled way, in the sense that they only describe a few techniques. The Cyc book provides some details about the implementation language CycL and it also provides some guidelines to distinguish the most important categories, but the concrete techniques for building Cyc are not detailed in the book. Uschold and King, Grüninger and Fox, Bernaras and her colleagues, SENSUS, and CO4 methodologies do not precisely describe the techniques and activities. (KA)^2 does not have public documentation describing in detail the

techniques used. METHONTOLOGY and the methodology for re-engineering have detailed descriptions for different techniques; however, some techniques remain to be detailed.

8. **There is a starting point to solve the maturity problems**. In this respect, this paper may be useful as a preliminary guide for ascertaining what the shortcomings are that should be overcome. Additionally, as in the case of METHONTOLOGY, existing standards for traditional software development can be used as guidelines.

References

Aguado, G, Bañón, A, Bateman, J, Bernardos, S, Fernández, M, Gómez-Pérez, A, Nieto, E, Olalla, A, Plaza, R and Sáchez, A, 1998, "ONTOGENERATION: reusing domain and linguisitic ontologies for Spanish text generation" *Workshop on Applications of Ontologies and Problem-Solving Methods*, European Conference of Artificial Ingelligence (ECAI) 1–10 (`http://delicias.dia.fi.upm.es/WORKSHOP/ECAI98/accepted-papers.html`).

Arpírez, JC, Corcho, O, Fernández-López, M and Gómez-Pérez, A, 2001, "WebODE: a workbench for ontological engineering". *First International Conference on Knowledge Capture* (K-CAP'01) 6–13 (`http://delicias.dia.fi.upm.es/webODE/Documents/arpirezEtAl01_submission.pdf`).

Arpírez, JC, Gómez-Pérez, A, Lozano, A and Pinto, HS, 2000, "Reference ontology and (ONTO)2Agent: the ontology yellow pages" *Knowledge and Information Systems*. **2**(2) 387–412 (`http://link.springer-ny.com/link/service/journals/10115/tocs/t0002004.htm`).

Bateman, J, Magnini, B and Fabris, B, 1995, "The generalized upper model knowledge base: organization and use" in N Mars (ed.) *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing* IOS Press.

Bateman, JA, 1994, "KPML: The KOMET-Penman (Multilingual) development environment" Technical Report, GMD/IPSI.

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

Bernaras, A, Laresgoiti, I and Corera, J, 1996, "Building and reusing ontologies for electrical network applications" *Proceedings of the European Conference on Artificial Intelligence* (ECAI'96) 298–302.

Blázquez, M, Fernández-López, M, García-Pinar, JM and Gómez-Pérez, A, 1998, "Building ontologies at the knowledge level using the ontology design environment" *Knowledge Acquisition of Knowledge-Based Systems Workshop* (KAW) SHARE 4–1–SHARE 4–15 (`http://ksi.cpsc.ucalgary.ca/KAW/KAW98/blazquez/`).

Ceccaroni, L, Cortés, U and Sánchez-Marré, M, 2000, "WaWO – an ontology embedded into an environmental decision-support system for wastewater treatment plant management" *Workshop on Applications of Ontologies and Problem solving Methods. 14th European Conference on Artificial Intelligence* (ECAI'00) 2–1–2–9 (`http://delicias.dia.fi.upm.es/WORKSHOP/ECAI00/accepted-papers.html`).

Chikofsky, EJ and Cross II, JH, 1990, "Reverse engineering and design recovery: a taxonomy" *Software Magazine* **7**(1) 13–17.

Downs, E, Clare, P and Coe, I, 1998, *Structured Analysis and Design Method (SSADM)* Prentice Hall.

Euzenat, J, 1995, "Building consensual knowledge bases: context and architecture" in N Mars (ed.) *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing* IOS Press.

Euzenat, J, 1996, "Corporative memory through cooperative creation of knowledge bases and hyper-documents" *Proceedings of the 10th KAW* (`http://ksi.cpsc.ucalgary.ca/KAW/KAW96/euzenat/euzenat96b.html`).

Farquhar, A, Fikes, R and Rice, J, 1997, "The Ontolingua server: a tool for collaborative ontology construction". *International Journal of Human Computer Studies*. **46**(6) 707–728.

Fensel, D, Decker, S, Erdman M and Studer, R, 1998, "Ontobroker: the very high idea" *Proceedings of the 11th International Flairs Conference* (FLAIRS-98) 131–135 `http://citeseer.nj.nec.com/192612.html`).

Fernández, M, Gómez-Pérez, A and Juristo, N, 1997, "METHONTOLOGY: from ontological art towards ontological engineering" *AAAI Spring Symposium on Ontological Engineering* 33–40 (`http://delicias.dia.fi.upm.es/miembros/ASUN/SSS97.ps`)

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

Fernández-López, M, Gómez Pérez, A and Rojas Amaya, MD, 2000, "Ontologies' crossed life cycles" *International Conference on Knowledge Engineering and Knowledge Management* (EKAW) 65–79.

Genesereth, MR and Fikes, RE, 1992, *Knowledge Interchange Format. Version 3.0. Reference Manual*. Computer Science Department. Stanford University, Stanford, California.

Goasdoué, F, Lattes, V and Rousset, MC, 2000, "The use of Carin language and algorithms for information integration: the Picsel system" *International Journal on Cooperative Information Systems* **9**(4) 383–401 (`http://www.lri.fr/ mcr/publis/ijcis.ps`).

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

Gómez-Pérez, A, 1998, "Knowledge sharing and reuse" in J Liebowitz (ed.) *Handbook of Expert Systems* CRC.

Gómez-Pérez, A, Fernández-López, M and de Vicente, A, 1996, "Towards a method to conceptualize domain ontologies" *Workshop on Ontological Engineering. European Conference of Artificial Intelligence* (ECAI) 41–52. (`http://delicias.dia.fi.upm.es/miembros/ASUN/ECAI96.ps`).

Gómez-Pérez, A, Juristo, N, Montes, C and Pazos, J, 1997, *Ingeniería del Conocimiento* Ed. Centro de Estudios Ramón Areces.

Gómez-Pérez, A, Juristo, N and Pazos, J, 1995, "Evaluation and assessment of knowledge sharing technology" in NJ Mars (ed.) *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing. KBKS 95* IOS Press, 284–296.

Gómez-Pérez, A and Rojas, MD, 1999, "Ontological reengineering and reuse" *International Conference on Knowledge Engineering and Knowledge Management* (EKAW) 139–156.

Gruber, T and Olsen, G, 1994, "An ontology for engineering mathematics" *Fourth International Conference on Principles of Knowledge Representation and Reasoning*. 258–269 (`http://www-ksl.stanford.edu/knowledge-sharing/papers/engmath.html`).

Grüninger, M and Fox, MS, 1995, "Methodology for the design and evaluation of ontologies" *Workshop on Basic Ontological Issues in Knowledge Sharing* (`http://www.eil.utoronto.ca/enterprise-modelling/papers/gruninger-ijcai95.pdf`) 6-1 to 6-10.

Guarino, N and Welty, C, 2000, "A formal ontology of properties" *Proceedings of EKAW-2000: The 12th International Conference on Knowledge Engineering and Knowledge Management* 97–112 (`http://www.ladseb.pd.cnr.it/infor/Ontology/Papers/OntologyPapers.html`).

Guarino, N and Welty, C, 2001, "Identity and subsumption" in Rebecca Green, Carol A Bean and Sung Hyon Myaeng (eds.), *The Semantics of Relationships: An Interdisciplinary Perspective*Kluwer (`http://www.ladseb.pd.cnr.it/infor/Ontology/Papers/OntologyPapers.html`).

IEEE, 1990, *IEEE Standard Glossary of Software Engineering Terminology.* Std. 610.12–1990 IEEE Computer Society.

IEEE, 1995, *IEEE Guide for Software Quality Assurance Planning*. Std. 730.1–1995 IEEE Computer Society.

IEEE, 1996, *IEEE Standard for Developing Software Life Cycle Processes*. Std. 1074–1995 IEEE Computer Society.

KACTUS, 1996, *The KACTUS Booklet Version 1.0* Esprit Project 8145. (`http://www.swi.psy.uva.nl/projects/ NewKACTUS/Reports.html`)

Kankaanpää, T, 1999, "Design and implementation of a conceptual network and ontology editor" VTT Information Technology, Research report TTE1–4–99 (`http://www.vtt.fi/tte/projects/language_engineering/publications/thesis.pdf`).

Kendall, KE and Kendall, JE, 1995, *Systems Analysis and Design* 3rd edition Prentice Hall.

Kendall, KE and Kendall, JE, 2002, *Systems Analysis and Design* 5th edition Prentice Hall.

Kifer, M, Lausen, G and Wu, J, 1995, "Logical foundations of object-oriented and frame-based languages" *Journal of the ACM* **42**(3) 741–843

Knight, K, Chancer, I, Haines, M, Hatzivassiloglou, V, Hovy, EH, Iida, M, Luk, SK, Whitney, RA and Yamada, K, 1995, "Filling knowledge gaps in a broad-coverage MT system" *Proceedings of the 14th IJCAI Conference* 1390–1396.

Knight, K and Luck S, 1994, "Building a large knowledge base for machine translation" *Proceedings of the American Association of Artificial Intelligence Conference* (AAAI-94) Vol. 1 pp. 773–778.

Leger, A, Arbaut, G, Barrett, P, Gitton, S, Gómez-Pérez, A, Holm, R, Lehtola, A, Mougenot, I, Nistal, A, Varvarigou, T and Vinesse, J, 2000. "Ontology domain modeling support for multi-lingual services in E-Commerce: MKBEEM". *ECAI'00 Workshop on Applications of Ontologies and PSMs* (`http://delicias.dia.fi.upm.es/WORKSHOP/ECAI00/accepted-papers.html`). Pages 19-1 to 19-4.

Lenat, DB and Guha, RV, 1990, *Building Large knowledge-Based systems* Addison-Wesley.

MAP, 1990, *Metodología de Planificación y Desarrollo de Sistemas de Información* Métrica versión 2. Ministerio para las Administraciones Públicas (MAP).

McCracken, DD, Jackson, MA, 1982, "Life cycle concept considered harmful" *ACM Software Engineering Notes* **7**(2) 29–32.

MacGregor, RM, 1992, "LOOM users manual" *ISI/WP-22*. Information Sciences Institute (ISI).

Mohedano, E, 2000, "Ontología de parámetros químicos de iones monoatómicos (alcalinos e hidrógeno) en variables físicas y humanas para aplicaciones de medio ambiente" Proyecto Fin de Carrera, Facultad de Informática, Universidad Politécnica de Madrid.

Nirenburg, S and Defrise, C, 1992, “Application-oriented computational semantics” in R Johnson and M Rosner (eds.) *Computational Linguistics and Formal Semantics* Cambridge University Press.

Pinilla Ponz, P, 1999, “Ontología de minerales. Aplicación a la clase de los silicates” final-year project, Facultad de Informática de la Universidad Politécnica de Madrid.

Rechenman, F, 1993, “Building and sharing large knowledge bases in molecular genetics” in N Mars (ed.) *Building and Sharing Large Knowledge Bases* IOS Press (`ftp://ftp.inrialpes.fr/pub/sherpa/publications/rechenmann93.ps.gz`).

Rojas, MD, 1998, “Ontologías de iones monoatómicos en variables físicos del medio ambiente” final-year project, Facultad de Informática de la Universidad Politécnica de Madrid.

Royce WM, 1987, “Managing the development of large software systems” *Proceedings 9th International Conference Software Engineering* IEEE Computer Society 328–338.

Schreiber, G, Wielinga, B and Jansweijer W, 1995, “The KACTUS view on the ‘O’ world” *IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing; Proceedings of the 7th Dutch National Conference on Artificial Intelligence NAIC’95* 159–168 (`http://www.swi.psy.uva.nl/usr/Schreiber/papers/Schreiber95a.pdf`) 15-1 to 15-10.

Studer, R, Benjamins, R and Fensel, D, 1998, “Knowledge engineering: principles and methods” *Data and Knowledge Engineering* **25**(1–2) 161–197.

Swartout, B, Ramesh P, Knight, K and Russ, T, “Toward distributed use of large-scale ontologies” *Symposium on Ontological Engineering of AAAI* 138–148.

Uschold, M, 1996, “Building ontologies: towards a unified methodology” *16th Annual Conference of the British Computer Society Specialist Group on Expert Systems* (`http://citeseer.nj.nec.com/uschold96building.html`).

Uschold, M and Grüninger, M, 1996, “Ontologies: principles methods and applications” *Knowledge Engineering Review* **11**(2) 93–137. (`http://citeseer.nj.nec.com/uschold96ontologie.html`).

Uschold, M and King, M, 1995, “Towards a methodology for building ontologies” *Workshop Held in Conjunction with IJCAI on Basic Ontological Issues in Knowledge Sharing* (`http://citeseer.nj.nec.com/uschold95toward.html`) 19-1 to 19-12.

Valente, A, Russ, T, McGregor, R and Swartout, W, 1999, “Building and (re)using an ontology of air campaign planning” *IEEE Intelligent Systems and their Applications* **4**(1) 27–36.

Waterman, DA, 1986, *A Guide to Expert Systems* Addison-Wesley.

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

Wielinga, BJ, Schreiber, AT and Breuker, JA, 1992, “KADS: a modeling approach to knowledge engineering” *Knowledge Acquisition* **4**(1) 5–53.