Overview Of Methodologies For Building Ontologies

Fernández López, M.
Laboratorio de Inteligencia Artificial
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo sn.
Boadilla del Monte, 28660. Madrid, Spain.
Tel: (34-91) 336-74-39,
Fax: (34-91) 336-7412
Email: {mfernand}@delicias.dia.fi.upm.es

Abstract

A few research groups are now proposing a series of steps and methodologies for developing ontologies. However, mainly due to the fact that Ontological Engineering is still a relatively immature discipline, each work group employs its own methodology. Our goal is to present the most representative methodologies used in ontology development and to perform an analysis of such methodologies against the same framework of reference. So, the goal of this paper is not to provide new insights about methodologies, but to put it all in one place and help people to select which methodology to use.

1 Introduction

One important difference between a technical field that is in its "infancy" and another that has reached "adulthood" is that the mature field has widely accepted methodologies, while the emerging discipline usually does not. Software Engineering, for example, can be said to have reached adulthood, because it has widely accepted methodologies; indeed, although different development methodologies are used in the United Kingdom and Spain -SSADM [Dow98] and Métrica 2 [MAP90] respectively-, both adopt the same principles and viewpoints and provide similar activities and techniques. At the Knowledge Engineering field, also exist methodologies: Common Kads [Wie92], Waterman's methodology [Wat86], IDEAL [Góm97], etc. With regard to Ontological Engineering, we believe that it is important to know both the state of the art of methodologies for ontology development and what problems need to be solved before the methodologies can be considered mature and can be applied with satisfactory prospects of success. We, therefore, consider that there is a need to diagnose the state of the art of methodologies for ontology development, and we will present and analyse the best known methodologies today against the *IEEE Standard for Developing Software Life Cycle Processes*, 1074-1995 [IEE96].

Our study will be structured as follows:

- *Section 2*. **IEEE Standard 1074-1995.** The standard document will be briefly described and we will discuss to what extent any ontology development methodology should comply with the provisions of this document.
- *Section 3*. **Brief history of methodologies**. We will discuss, chronologically, how a series of methodologies have evolved from 1995 to the present day.
- *Section 4*. **Criteria for analysing methodologies**. We will describe the generic characteristics that will be taken as a reference point for conducting the comparative study of the methodologies.
- *Sections 5 to 9*. **Analysis of each methodology.** We will examine the above methodologies by means of: a discussion of each methodology, an analysis of

The copyright of this paper belongs to the papers authors. Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage.

Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5) Stockholm, Sweden, August 2, 1999

(V.R. Benjamins, B. Chandrasekaran, A. Gomez-Perez, N. Guarino, M.
Uschold, eds.)

http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-18/

each methodology according to the criteria established in section 4 and a summary. The methodologies to be studied will be Uschold's methodology [Usc95], [Usc96], [UsG96], Grüninger's methodology [Grü95], [Usc96], [UsG96], the proposal by Bernaras et al. [Ber96], METHONTOLOGY [Góm96], [Fer97], [Góm98], [Fer99] and the methodology used in SENSUS [Swa97].

- *Section 10.* **Conclusions: summary of the methodology analysis.** A series of general conclusions will be drawn from the analysis.

2 IEEE Standard 1074-1995

2.1 Description

The IEEE 1074-1995 standard [IEE96] 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 time 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:

1. **Software life cycle model process:** 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.

2. **Project management processes**: 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.

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

   3.1. **Pre-development processes**: are performed before starting software development proper. They involve activities related to studying the environment in which the software is to operate and conducting feasibility studies.

   3.2. **Development processes**: are processes that must be performed to build the software product. These processes include:

   - **Requirements process**: includes iterative activities directed towards developing the software requirements specification.
   - **Design process**: its goal is to develop a coherent and well-organized representation of the software system that meets the requirements specification.
   - **Implementation process**: Transforms the design representation of a software product into a programming language realization.

   3.3. **Post-development processes**: are related to the installation, operation, support, maintenance and retirement of a software product. They are performed after software construction.

4. **Integral processes**: are needed to successfully complete software project activities. They ensure the completion and quality of project functions. They are carried out 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.

2.2 Why and How Can the IEEE Standard Be Applied To Ontology Development

According to the IEEE definition [IEE90], 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 proposed for software generally, which should be adapted to the special characteristics of ontologies. Below, we describe to what extent the IEEE standard processes should be applied in an ontology development methodology:

1. **Software life cycle model process**. The methodology should recommend one or more life cycles from which the developer can select one.

2. **Project management processes**. The activities proposed by the standard for these processes are applicable to any software product and it is, therefore, recommendable that they be applied in ontology development.

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

   3.1. **Pre-development processes.** Apart from studying the 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.

3.2. **Development processes.** By process types, it can be said that:

- **Requirements process.** As shown in [Góm98], ontologists are able to specify, at least partially, what is expected of the ontology.
- **Design process.** Ontologies also have to be designed, albeit differently from other types of software. According to the philosophy of the standard, it is not recommendable to go directly from requirements specification to coding.
- **Implementation process.** Obviously, if ontologies are to be used by computers, they have to be implemented.

3.3. **Post-development processes.** Are activities that are common to any type of software.

4. **Integral processes**. The activities proposed by the standard for these processes can be applied to any type of software. This includes training, since the personnel responsible for maintaining the ontologies, for example, need instruction.

3 Brief History Of The Methodologies

On the basis of the experience gathered in developing the *Enterprise Ontology* [Usc95] and the TOVE (*TOronto Virtual Enterprise*) project ontology [Grü95] (both in the domain of enterprise modelling), the first methodological outlines were proposed in 1995 and later refined in [Usc96] and [UsG96]. At the 12th European Conference for Artificial Intelligence (ECAI'96) held in 1996, Bernaras et al. [Ber96] presented a method used to build an ontology in the domain of electrical networks as part of the Esprit KACTUS project. METHONTOLOGY [Góm96] appeared at the same time and was extended in later papers [Fer97], [Góm98], [Fer99]. In 1997, that is, one year later, a methodology was proposed for building ontologies based on the SENSUS ontology [Swa97].

The methodology proposed by Uschold will be described in section 5; Grüninger's methodology will be discussed in section 6, the methodology proposed by Bernaras et al. in section 7, METHONTOLOGY in section 8, and finally, the SENSUS methodology in section 9.

4 Criteria For Analysing Methodologies

The criteria that we have established for analysing each methodology are:

C1. **Inheritance from Knowledge Engineering**. Consideration of the influence of traditional Knowledge Engineering on the methodology in question.

C2. **Detail of the methodology.** Consideration of whether the activities and techniques proposed by the methodology are exactly specified.

C3. **Recommendations for knowledge formalization.** Consideration of the formalism or formalisms proposed for representing knowledge (logic, frames, etc.).

C4. **Strategy for building ontologies**. Discussion of which of the following strategies are used to develop ontologies:

a. **Application-dependent**: the ontology is built on the basis of an application knowledge base, by means of a process of abstraction.

b. **Application-semidependent**: possible scenarios of ontology use are identified in the specification stage.

c. **Application-independent**: the process is totally independent of the uses to which the ontology will be put in knowledge-based systems, agents, etc.

C5. **Strategy for identifying concepts.** The possible strategies are [UsG96]: 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).

C6. **Recommended life cycle**. Analysis of whether the methodology implicitly or explicitly proposes a life cycle.

C7. **Differences between the methodology and IEEE 1074-1995.** Discussion of which of the processes and activities proposed by the IEEE standard 1074-1995 are not mentioned in the methodology.

C8. **Recommended techniques.** Specification of whether particular techniques are proposed for performing the different activities of which the methodology is composed.

C9. **What ontologies have been developed using the methodology and what systems have been built using these ontologies.** The ontologies and systems developed will be listed and briefly described.

Criteria C1, C2, C3, C4 and C5 will show general points of the methodologies. The other criteria will show the maturity of each methodology. Another interesting criterion for analysis would be collaborative and distributive construction, that is, to what extent the methodologies permit different groups at different sites to work together to build ontologies; however, none of the publications to date mention what each one

contributes in this respect.

5 Methodology By Uschold And King

5.1 Description

This methodology is based on the experience of developing the *Enterprise Ontology*, an ontology for enterprise modelling processes [Usc95]. This methodology provides guidelines for developing ontologies, which are:

1. **Identify purpose.** It is important to be clear 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**, which means:
   - Identification of the key concepts and relationships in the domain of interest, that is, scoping. It is important to centre on the concepts as such, rather than the words representing them.
   - Production of precise unambiguous text definitions for such concepts and relationships.
   - Identification of terms to refer to such concepts and relationships.

   Agreeing on all of the above.

   The authors use a middle-out approach to perform this step and recommend that rather than looking for the most general or the most particular concepts as key concepts, the most important concepts be identified, which will then be used to obtain the remainder of the hierarchy by generalization and specialization.

   2.2. **Coding.** Involves explicitly representing the knowledge acquired in step 2.1 in a formal language.

   2.3. **Integrating existing ontologies.** During either or both of the capture and coding processes, there is the question of how and whether to use ontologies that already exist.
3. **Evaluation**, where the authors adopt the definition of [Góm95]: "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.

5.2 Ontologies Developed Using This Methodology

The most important project 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 Artificial Intelligence Applications Institute at the University of Edinburgh with its partners: IBM, Lloyd's Register, Logica UK Limited, and Unilever. (See *http://www.aiai.ed.ac.uk/project/enterprise/enterprise/ontology.html*)

5.3 Systems Built Using Ontologies Developed With This Methodology

The most important tool developed using the Enterprise Ontology is the **Enterprise Toolset.** It 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, visualization, and support for process enactment, and an Enterprise Ontology for communication. (See *http://www.aiai.ed.ac.uk/project/enterprise* for more information).

5.4 Analysis Of The Methodology

According to the criteria established in section 4, the following can be said:

C1. **Inheritance from Knowledge Engineering**. This methodology recalls knowledge-based systems development in the sense that it clearly identifies an acquisition, coding and evaluation stage. However, it proposes neither a feasibility study nor prototyping, thereby differentiating it from knowledge-based systems development.

C2. **Detail of the methodology.** Little, it does not precisely describe the techniques and activities.

C3. **Recommendations for knowledge formalization.** None in particular.

C4. **Strategy for building applications**. The process is totally independent of the uses to which the ontology will be put and is, therefore, application-independent

C5. **Strategy for identifying concepts**. The key concepts are established by searching first for the most important, rather than the most general or

most particular concepts; the others are obtained by generalization and by specialization. Therefore, a middle-out strategy can be said to be used for identifying concepts.

C6. **Recommended life cycle.** This methodology does not propose a life cycle.

C7. **Differences between the methodology and IEEE 1074-1995.** As shown in table 1, it does not mention management, pre-development and post-development, nor does it propose a design process. Some activities are missing in the processes it does propose (requirements, implementation and integral processes), particularly: environment study, feasibility study, training and configuration management.

C8. **Recommended techniques.** Techniques for performing the different activities are not given in detail. For example, the methodology recommends that the key concepts and relationships in the domain under study be identified during acquisition; however, no details are given about how this should be done, and only a very vague guideline, involving the use of brainstorming techniques, is given.

C9. **What ontologies have been developed using the methodology and what systems have been built using these ontologies**. Complex projects have been developed on business domains, which have served to validate the feasibility of the methodology.

6 Methodology By Grüninger And Fox

6.1 Description

This methodology is based on the experience in developing the TOVE project ontology [Grü95] 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 constructed directly. First, an informal description is made of the specifications to be met by the ontology and then this description is formalized. The steps proposed are as follows:

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 characterize the answers to these questions using the axioms and definitions. These are the informal competency questions, since they are not yet expressed in the 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 will be taken:

   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 formalism such as KIF [Gen92]. 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 posed informally and the terminology of the ontology has been defined, the competency questions are defined

formally.

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 on 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 characterize 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 characterizing 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.

6.2 Ontologies Developed Using This Methodology

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 formalized using first-order logic. The TOVE ontologies include: Enterprise Design Ontology, Project Ontology, Scheduling Ontology, or Service Ontology. For more information, see *http://www.ie.utoronto.ca/EIL*.

6.3 Applications Using Ontologies Developed With This Methodology

The ontologies built according to this methodology have been used in the applications listed below:

1. **Enterprise Design Workbench.** It 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 re-design, where the workbench provides a comparative analysis of enterprise design alternatives, and guidance to the designer.

2. **Integrated Supply Chain Management Project agents.** The goal is to organize 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: logistic, transportation, management, etc.

The applications described in this section are still under development. For further information, see *http://www.ie.utoronto.ca/EIL*.

6.4 Analysis Of The Methodology

According to the criteria in section 4, the following can be said:

C1. **Inheritance from Knowledge Engineering.** As is usual practice in knowledge-based systems development, this methodology identifies questions, which, in this case, the ontology must be capable of answering; however, there is no clear-cut division into the stages involved in knowledge-based systems development.

C2. **Detail of the methodology.** Neither the activities nor the techniques are described in detail.

C3. **Recommendations for knowledge formalization**. Clearly opts for logic.

C4. **Strategy for building applications.** Ontology use scenarios are identified in the specification stage, so it is a application-semidependent strategy.

C5. **Strategy for identifying concepts.** It adopts a middle-out strategy.

C6. **Recommended life cycle**. No life cycle model selection process is identified, nor is any explicit reference made to there being any preference for one model over another; 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, there is no statement concerning whether or not the definitions it already contains can be modified when extending an ontology. Accordingly, it is impossible to ascertain whether the methodology admits development by means of evolving prototypes or only an incremental life cycle.

C7. **Differences between the methodology and IEEE 1074-1995.** As shown in table 1, neither design nor management, pre-development and post-development processes are proposed and, for the processes that are mentioned (requirements, implementation and integral processes), no reference is made to the activities concerning: training and configuration management.

C8. **Recommended techniques.** There is no detailed description of techniques, for example, the techniques for formulating the competency

questions are not mentioned.

C9. **What ontologies have been developed using the methodology and what systems have been built using these ontologies.** Complex projects have been developed; albeit all in the same domain.

7 The approach of Amaya Berneras et al.

7.1 Description

The work of Bernaras et al. is set within the Esprit KACTUS project [KAC96]. One of the objectives of the KACTUS project is to investigate the feasibility of knowledge reuse in complex technical systems and the role of ontologies to support it [Sch95].

This approach to developing ontologies is conditioned by applications development. So, every time an application is built, the ontology that represents the knowledge required for the application is built. This ontology can be developed by reusing others and can also be integrated into the ontologies of later applications. Therefore, every time an application is developed, the following steps are taken:

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 the top-level ontological categories determined.

   This design process involves searching ontologies developed for other applications, which are refined and extended for use in the new application.

3. **Ontology refinement and structuring** in order to arrive at a definitive design. The principles of minimum coupling can be used to assure that the modules are not very dependent on each other and are as coherent as possible, looking to get maximum homogeneity within each module.

7.2 Ontologies And Applications Developed With This Methodology

As experience based on this approach, 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 and the third controls the electrical network on the basis of the above two applications.

7.3 Analysis Of The Methodology

According to the criteria set out in section 4, the following can be said:

C1. **Inheritance from Knowledge Engineering.** This method follows in the tradition of knowledge engineering. Indeed, it considers the construction of ontologies at the same time as knowledge-based system development.

C2. **Detail of the methodology**. Very little.

C3. **Recommendations for knowledge formalization.** None.

C4. **Strategy for building applications.** The construction of ontologies is based on the construction of particular applications. As more applications are built, the ontology becomes more general, and, therefore, moves further away from what would be a traditional knowledge base. So, this methodology can be said to follow an application-dependent strategy in this respect.

C5. **Strategy for identifying concepts.** Top-down.

C6. **Recommended life cycle**. It simply seems to assume that the life cycle should be the same as is used in the development of the application associated with the ontology.

C7. **Differences between the methodology and IEEE 1074-1995.** The management, pre-development and post-development processes are missing. Also, for the integral processes, training, documentation, configuration management, verification and validation are missing.

C8. **Recommended techniques.** No particular techniques are described.

C9. **What ontologies have been developed using the methodology and what systems have been built using these ontologies.** The methodology has been used in the domain of electrical networks.

8 METHONTOLOGY

This methodology was developed within the Laboratory of Artificial Intelligence at the Polytechnic University of Madrid. The METHONTOLOGY framework [Fer97], [Góm98], [Fer99] enables the construction of ontologies at the knowledge level and includes [Bláz98]: the identification of the ontology development process, a life cycle based on evolving prototypes, and particular techniques for carrying out each activity. The METHONTOLOGY framework is supported by ODE [Bláz98], [Fer99].

8.1 Ontology Development Process

The ontology development process [Fer97] refers to *which* activities are carried out 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 co-operative teams with some assurance of correctness and completeness. If this is the case, it is advisable to perform the three categories of activities presented below and steer clear of anarchic constructions.

Project Management Activities include planning, control and quality assurance. *Planning,* identifies which tasks are to be performed, how they will be arranged, how much time and what resources are needed for their completion. This activity is essential for ontologies that need to use ontologies which have already been built or ontologies that require levels of abstraction and generality. *Control*, 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 outputted (ontology, software and documentation) is satisfactory. [Roj98] describes how these activities are performed.

Development-Oriented Activities include specification, conceptualization, formalization and implementation. *Specification* states why the ontology is being built and what are its intended uses and who are the end-users. *Conceptualization* structures the domain knowledge as meaningful models at the knowledge level. *Formalization* 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. [Fer99] gives details of how all the development activities, except Formalization and Maintenance, are performed.

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 [Góm95]. *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 phases completed and products generated. *Configuration Management* records all the versions of the documentation, software and ontology code to control the changes. In [Fer99], [GóR99], a description is given of how Knowledge Acquisition was performed in the CHEMICALS ontology, and Evaluation, Integration and Configuration Management is discussed in [Roj98], where the documentation outputted is discussed as part of the description of each activity.

8.2 Ontology Life Cycle

It identifies the *set of stages* through which the ontology moves during its life time, describes what activities are to be performed in each stage and how the stages are related (relation of precedence, return, etc.). In [Fer97], a justification is given of why the ontology life cycle should be based on evolving prototypes.

8.3 Ontologies Developed With This Methodology

The most important ontologies built according to METHONTOLOGY are:

1. CHEMICALS [Fer96], [Góm96], [Fer99], which contains knowledge within the domain of chemical elements and crystalline structures.

2. Environmental pollutants ontologies [Roj98] [GóR99]. They represent the methods of detecting the different pollutant components of various media: water, air, soil, etc., and the maximum permitted concentrations of these components, taking into account all the legislation in force (European Union, Spanish, German, US regulations, etc.).

3. The Reference-Ontology [Arp98], 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 organization.

4. The restructured version of the $(KA)^2$ ontology [Bláz98], which contains knowledge about the scientific community in the field of Knowledge Acquisition, particularly: scientists, research topics, projects, universities, etc.

This methodology has been proposed for ontology construction by the Foundation for Intelligent Physical Agents (FIPA), which promotes inter-operability across agent-based applications (*http://www.fipa.org*).

8.4 Applications Using Ontologies Developed With This Methodology

The most important applications are shown below:

1. **$(Onto)^2$Agent** [Arp98]. 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. It is available at (*http://delicias.dia.fi.upm.es/ OntoAgent*).

2. **Chemical OntoAgent** [Arp98]. 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.

3. **Ontogeneration** [Agu98]. It is a system that uses a domain ontology (CHEMICALS) and a linguistic ontology (GUM [Bat95]) to generate Spanish text descriptions in response to the queries of students in the domain of chemistry.

8.5 Analysis Of The Methodology

According to the criteria set out in section 4, the following can be said:

C1. **Inheritance from Knowledge Engineering.** It has its roots in a methodology for developing Knowledge-Based Systems (IDEAL [Góm97]).

C2. **Detail of the methodology**. A sizable part of the methodology is very detailed; the remainder will be specified in more detail in the future.

C3. **Recommendations for knowledge formalization.** METHONTOLOGY gives freedom of choice with regard to formalization. If the ODE tool is used it is not even necessary, because ODE generates target codes from the ontology definition at the knowledge level.

C4. **Strategy for building applications.** Application-independent.

C5. **Strategy for identifying concepts.** The most relevant concepts are identified first, so it adopts a middle-out strategy.

C6. **Recommended life cycle.** Evolving prototypes.

C7. **Differences between the methodology and IEEE 1074-1995.** The process groups that are missing are: software life cycle model process, although an evolving prototype-based life cycle is proposed for any ontology developed, and pre-development processes. For the other processes (project management processes, development-oriented processes and integral processes), the following proposals are missing: project initiation, installation, support, retirement and training.

The similarity between METHONTOLOGY and the IEEE standard is due to the fact that the skeleton of METHONTOLOGY was developed taking this document as a starting point.

C8. **Recommended techniques.** Techniques for the Control activity remain to be specified.

C9. **What ontologies have been developed using the methodology and what systems have been built using these ontologies.** It has been used to develop ontologies and applications in different domains.

9 SENSUS-Based Methodology

9.1 The SENSUS Ontology

This is an ontology for use in natural language processing and was developed at the ISI (Information Sciences Institute) natural language group to provide a broad-based conceptual structure for developing machine translators [Kni94] [Kni95]. Its current content was obtained by extracting and merging information from various electronic sources of knowledge. This process began by merging the PENMAN Upper Model [Bat89] and ONTOS (two, very high level linguistically-based ontologies) and the semantic categories from a dictionary by hand to produce an 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. After, to support machine translation, the result of this merge was then augmented by Spanish and Japanese lexical entries from the Collins Spanish/English dictionary and the Kenkyusha Japanese/English dictionary [Swa97].

SENSUS has more than 50,000 concepts organized 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, does not cover terms from specific domains. The domain terms are linked with SENSUS in order to build ontologies for particular domains, and any irrelevant terms are pruned in SENSUS.

9.2 The Methodology According To The SENSUS Approach

When an ontology is to be built in a particular domain, the following steps are taken [Swa97]:

1. A series of terms are taken as *seed*.
2. These seed terms are linked by hand to SENSUS.
3. All the concepts in the path from the seed 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.

9.3 Ontologies Developed Using This Methodology

An ontology for military air campaign planning has been built using SENSUS. It contains an overview of the basic elements that characterize air campaign plans, such as campaign, scenario, participants, commanders, etc. [Val99]. This ontology includes ontologies on weapons, systems in general, fuel, etc.

9.4 Applications Using Ontologies Developed With This Methodology

On the basis of SENSUS, knowledge-based applications for the air campaign planning domain have been developed at ISI in conjunction first with the ARPA Rome Planning Institute program and later with the DARPA Joint Forces Air Component Commander program. These include the Strategy Development Assistant [Val99], a tool that provides support for intelligent and guided plan development.

9.5 Analysis Of The Methodology

The following can be said:

C1. **Inheritance from Knowledge Engineering**. It completely breaks with the tradition, as it is based on adding terms to an existing ontology (SENSUS), which is then pruned.

C2. **Detail of the methodology.** It is not very detailed.

C3. **Recommendations for knowledge formalization**. Semantic networks.

C4. **Strategy for building applications.** Application-semidependent, as the seed terms are obtained with an application in mind.

C5. **Strategy for identifying concepts**. It is bottom-up, as first the most specific concepts required for the application are sought and, then, the search-and-prune method is applied to enter more abstract concepts.

C6. **Recommended life cycle**. No preference for a particular model is stated, since nothing is said about how to develop versions other than the first.

C7. **Differences between the methodology and IEEE 1074-1995**. The design process and the management, pre-development and post-development processes are missing. For the integral processes, the activities related to training, documentation, configuration management, verification and validation are missing.

C8. **Recommended techniques**. No particular techniques are detailed.

C9. **What ontologies have been developed using the methodology and what systems have been built using these ontologies.** Both the ontologies and the applications centre on the military campaigns domain.

10 Conclusions: Summary Of The Analysis Of Methodologies

According to the above analysis, recapitulated in table 2, 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. Additionally, it is recommended by the FIPA.

   ii. Grüninger and Fox's methodology, which includes neither the processes described in section 4, nor activities and techniques for performing such activities. 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, which, apart from the shortcomings of the above methodologies, does not mention the life cycle.

   v. Bernaras et al.'s methodology, which, apart from the above omissions, has not been used to build many ontologies and applications.

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 [Usc96], cited in section 3. Its disadvantage was that the new synthesized methodology was not an actual methodology, it was a conception of a potential methodology.

3. **There is one aproach that is completely different from the others: SENSUS**. This may mean that the best we can do is to have several widely accepted

methodologies rather than just one standarized methodology.

4. **It is allowed interoperatibilty between systems**. Domain ontologies built using SENSUS approach share the same high level concepts (or skeleton). So, systems that use such ontologies will share a common structure of the world, and it would be aesier for them to communicate because the share the same underlying structure.

5. **There is a starting point for solving the above problems.** We have a series of methodologies that can be used as a reference point for developing one or several standardized methodologies adaptable to different ontology types in different settings. In this respect, this paper may be useful as preliminary guide for ascertaining what are the shortcoming of existing methodologies that should be overcome by futur methodologies. Additionally, as in the case of methodology, existing standards for traditional software development can be used as guidelines.

Acknowledgements

I would like to thank Asunción Gómez Pérez for her help on this paper.

References

[Agu98] Aguado, G.; Bañón, A.; Bateman, J.; Bernardos, S.; Fernández, M.; Gómez-Pérez, A. Nieto, E.; Olalla, A.; Plaza, R.; Sáchez, A. **ONTOGENERATION: Reusing domain and linguistic ontologies for Spanish text generation**. Workshop on Applications of Ontologies and Problem-Solving Methods. European Conference on Artificial Intelligence (ECAI'98). Brighton (United Kingdom). 1998.

[Arp98] Arpírez, J. C.; Gómez-Pérez, A.; Lozano, A. Pinto, H. S. **Reference Ontology and (ONTO)^{2}Agent: The ontology yellow pages**. Workshop on Applications of Ontologies and Problem-Solving Methods. European Conference on Artificial Intelligence (ECAI'98). Brighton (United Kingdom). 1998.

[Bat95] Bateman, J.; Magnini, B.; Fabris, B. **The Generalized Upper Model Knowledge Base: Organization and Use.** In N. Mars (Editor). Towards Very Large Knowledge Bases. IOS Press. 1995. pp. 60-72.

[Bat89] Bateman, J. A.; Kasper, R. T.; Moore, J. D.; Whitney, R. A. **A General Organization of Knowledge for Natural Language Processing: The Penman Upper Model**. USC/Information Sciences Institute. Marina del Rey. 1989.

[Bern96] Bernaras, A.; Laresgoiti, I.; Corera, J. **Building and Reusing Ontologies for Electrical Network Applications** Proceedings of the European Conference on Artificial Intelligence (ECAI'96). ECAI 96. 1996.

[Bláz98] Blázquez, M.; Fernández-López, M.; García-Pinar, J.M.; Gómez-Pérez, A. **Building Ontologies at the Knowledge Level using the Ontology Design Environment**. Knowledge Acquisition of Knowledge-Based Systems Workshop (KAW). 1998.

[Dow98] Downs, E.; Clare, P.; Coe, I. **Structured Analysis and Design Method (SSADM).** Prentice Hall. 1998.

[Fer99] Fernández-López, M.; Gómez-Pérez, A.; Pazos-Sierra, A.; Pazos-Sierra, J. **Building a Chemical Ontology Using Methontology and the Ontology Design Environment**. IEEE Intelligent Systems & their applications. January/February 1999. PP. 37-46.

[Fern97] Fernández, M.; Gómez-Pérez, A.; Juristo, N. **METHONTOLOGY: From Ontological Art Towards Ontological Engineering**. Symposium on Ontological Engineering of AAAI. Stanford (California). March 1997.

[Fer96] Fernández, M. **CHEMICALS: ontología de elementos químicos**. Final-Year Project. Facultad de Informática de la Universidad Politécnica de Madrid. 1996.

[GóR99] Gómez-Pérez, A; Rojas, M. D. **Ontological Reengineering and Reuse**. European Knowledge Acquisition Workshop (EKAW). 1999.

[Góm98] Gómez Pérez, A. **Knowledge Sharing and Reuse.** In J. Liebowitz (Editor) *Handbook of Expert Systems*. CRC. 1998.

[Góm97] Gómez-Pérez, A.; Juristo, N.; Montes, C.; Pazos, **J. Ingeniería del Conocimiento**. Ceura 1997.

[Góm96] Gómez-Pérez, A. **A Framework to Verify Knowledge Sharing Technology.** Expert Systems with Application. Vol. 11, No. 4. pp. 519-529. 1996.

[Góm95] Gómez-Pérez, A.; Juristo, N.; Pazos, J. **Evaluation and assessment of knowledge sharing technology.** In N. J. Mars (editor), Towards Very Large Knowledge Bases. KBKS 95. IOS Press, Amsterdam, 1995. pp., 289-296.

[Grün94] Grüninger, M.; Fox, M. S. **The Role of**

Competency Questions in Enterprise Engineering. IFIP WG 5.7 Workshop on Benchmarking. Theory and Practice. Trondheim, Norway. 1994.

[Gen92] Genesereth, M. R.; Fikes, R. E. **Knowledge Interchange Format. Version 3.0. Reference Manual**. Computer Science Department. Stanford University. Stanford, California. 1992.

[IEE96] **IEEE Standard for Developing Software Life Cycle Processes**. IEEE Computer Society. New York (USA). April 26, 1996.

[IEE90] **IEEE Standard Glossary of Software Engineering Terminology**. IEEE Computer Society. New York (USA). 1990.

[KAC96] **The KACTUS Booklet version 1.0**. Esprit Project 8145. September, 1996. (*http://www.swi.psy.uva.nl/prjects/NewKACTUS/Reports.html*)

[Kni95] Knight, K.; Chancer, I.; Haines, M.; Hatzivassiloglou. V.; Hovy, E. H.; Iida M.; Luk, S.K.; Whitney, R.A.; Yamada, K. **Filling Knowledge Gaps in a Broad-Coverage MT System**. Proceedings of the 14th IJCAI Conference. Montreal (Canada). 1995.

[Kni94] Knight, K.; Luck S. **Building an Large Knowledge Base for Machine Translation**. Proceedings of the American Association of Artificial Intelligence Conference (AAAI-94). Seattle (USA). 1994.

[MAP90] **Metodología de Planificación y Desarrollo de Sistemas de Información. Métrica versión 2**. Ministerio para las Administraciones Públicas (MAP). 1990.

[Roj98] Rojas, M. D. **Ontologías de iones monoatómicos en variables físicos del medio ambiente**. Final-Year Project. Facultad de Informática de la Universidad Politécnica de Madrid. 1998.

[Sch95] Schreiber, G.; Wielinga, B.; Jansweijer W. **The KACTUS View on the 'O' World**. In Proceedings of the National Dutch AI Conference. NAIC'95. 1995.

[Swa97] Swartout, B.; Ramesh P.; Knight, K.; Russ, T. **Toward Distributed Use of Large-Scale Ontologies**. Symposium on Ontological Engineering of AAAI. Stanford (California). Mars, 1997.

[Usc96] Uschold, M. **Building Ontologies: Towards A Unified Methodology**. Expert Systems 96. Cambridge. 1996.

[Usc95] Uschold, M. King, M. **Towards a Methodology for Building Ontologies**. Workshop on Basic Ontological Issues in Knowledge Sharing. 1995.

[UsG96] Uschold, M.; Grüninger, M. **Ontologies: Principles Methods and Applications**. Knowledge Sharing and Review. Vol. 2. 1996.

[Val99] Valente, A.; Russ, T.; McGregor, R.; Swartout, W. **Building and (Re)Using an Ontology of Air Campaign Planning**. IEEE Intelligent Systems & their applications. January/February 1999.

[Wat86] Waterman, D. A. **A Guide to Expert Systems**. Addison-Wesley. Masachusets (USA). 1986.

[Wie92] Wielinga, B. J.; Schreiber, A. T.; Breuker, J. A. **KADS: a modeling approach to knowledge engineering**. Knowledge Acquisition (4). PP. 5-53. 1992.

Table 1. Methodology Compliance With IEEE 1074-1995

	Project management processes	Project development-oriented processes					Integral processes
		Pre-development Processes	Development process			Post-development processes	
			Requirements process	Design process	Implementation process		
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
Bernaras	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

Table 2. Summary of methodologies

	Inheritance from Knowledge Engineering	Detail of the methodology	Recommendations for formalization	Strategy for building applications	Strategy for identifying concepts	Recommended life cycle	Differences from IEEE 1074-1995	Recommended techniques	Ontologies and applications	Collaborative and distributive construction
Uschold y King	Partial	Very little	None in particular	Application-independent	Middle-out	None	- Processes missing - Activities missing	Not known	One domain only	Not documented
Grüninger y Fox	Small	Little	Logic	Application-semidependent	Middle-out	To be detailed	- Processes missing - Activities missing	Not known	One domain only	Not documented
Bernaras	Big	Very little	None	Application-dependent	Top-down	None	- Processes missing - Activities missing	Not known	One domain only	Not documented
METHONTOLOGY	Big	A lot	None	Application-independent	Middle-out	Evolving prototypes	-Pre-development process missing - Activities missing	Some activities missing	Several domains	Not documented
SENSUS	None	Medium	Semantic networks	Application-semidependent	Not specified	To be detailed	- Processes missing - Activities missing	Not known	Several domains	Not documented