METHONTOLOGY: from ontological art towards ontological engineering

Article · March 1997

Source: OAI

CITATIONS
1,732

READS
8,235

3 authors, including:

Mariano Fernández-López
No affiliation
95 PUBLICATIONS **8,963** CITATIONS
SEE PROFILE

Natalia Juristo
Universidad Politécnica de Madrid
278 PUBLICATIONS **9,753** CITATIONS
SEE PROFILE

METHONTOLOGY: From Ontological Art Towards Ontological Engineering

Mariano Fernández, Asunción Gómez-Pérez, Natalia Juristo

Laboratorio de Inteligencia Artificial
Facultad de Informática
Universidad Politécnica de Madrid
Campus de Montegancedo sn.
Boadilla del Monte, 28660. Madrid, Spain.
Tel: (34-1) 336-74-39, Fax: (34-1) 336-7412
Email: {mfernand, asun, natalia}@delicias.dia.fi.upm.es

Abstact

This paper does not pretend either to transform completely the ontological art in engineering or to enumerate exhaustively the complete set of works that has been reported in this area. Its goal is to clarify to readers interested in building ontologies from scratch, the activities they should perform and in which order, as well as the set of techniques to be used in each phase of the methodology. This paper only presents a set of activities that conform the ontology development process, a life cycle to build ontologies based in evolving prototypes, and METHONTOLOGY, a well-structured methodology used to build ontologies from scratch. This paper gathers the experience of the authors on building an ontology in the domain of chemicals.

1. Introduction

Until now, a big quantity of ontologies have been developed by different groups, under different approaches, and using different methods and techniques. However a few works have been published about how to proceed, showing the practices, design criteria, activities, methodologies, and tools used to build them. The consequence is clear, the *absences* of standardized activities, life cycles, and systematic methodologies as well as a set of well-defined design criteria, techniques and tools make the ontologies development a craft rather than an engineering activity. So, the art will became engineering when there exist a definition and standardization of a life cycle that goes from requirements definition to maintenance of the finished product, as well as methodologies and techniques that drive their development.

One of the main problems that Knowledge Engineers (KEs) have when building expert systems is the difficulty of getting a set of requirements for the system. Requirements specify how the system will behave. Since experts are not usually able to describe in a concrete and complete way how they behave in the application domain, it is hard for KEs to specify the future behavior of the Knowledge-Based Systems (KBS). So, KBS are usually built incrementally using evolving prototypes in which the deficiencies of the final product of each cycle can be used as the specification of the next prototype. In ontologies do not occur the same. Ontologies are built to be reused or shared anytime, anywhere, and independently of the behavior and domain of the application that uses them. So, ontologists should be able to specify, at least partially, a big portion of the needed vocabulary that the ontology will cover for a given domain. This is the main difference between ontologies and KBS development processes. Consequently, methodologies used to build KBSs can not completely be applied to build ontologies.

In the following sections, this paper presents a set of activities that conforms the ontology development process (section #2), a life cycle of ontologies (section #3) and METHONTOLOGY (section #4), a method to build ontologies from scratch.

2. Ontology Development Process

The ontology development process refers to *what* activities you need to carry out when building your ontologies. However, the ontology development process does not imply an order of execution of such activities. Its goal is to identify the list of activities to be completed. Usually verbs are used to refer to such activities. The ontology development process is composed of the following:

* Before building your ontology, you should **plan** the main tasks to be done, how they will be arranged, how much time you need to perform them and with which resources (people, software and hardware).
* As you do not plan a trip without knowing your destination (your goal), you should not start the

development of your ontology without knowing its purpose and scope. So, try to answer the questions: why this ontology is being built and what are its intended uses and end-users. Then, **specify** or write the answers in an ontology requirements specification document. As an example of an informal and formal specification of two independent ontologies in the domain of modeling enterprise, to cite the Enterprise ontology (by Uschold) and the TOVE ontology (by Gruninger). Although their outputs are almost the same (a set of terms), the degree of formality in writing their requirements specification documents are different. The Enterprise ontology has been specified in natural language and the TOVE ontology using a set of competence questions (Uschold & Gruninger 1996).

* Unless you wish to build a toy ontology or unless you are an expert in the domain, you will elicit knowledge using KBSs knowledge elicitation techniques. As a result, you should be able to list the sources of knowledge and give, a rough description of how you carried out the processes, and of the techniques you used. The most extensive work on capturing knowledge was reported by Uschold and Gruninger (Uschold & Gruninger 1996).

* Once you have acquired enough knowledge you **conceptualize** it in a conceptual model that describes the problem and its solution. A set of intermediate representations for conceptualizing a domain ontology of objects were presented by Gómez-Pérez and colleagues at (Gómez-Pérez, Fernández, & De Vicente 1996).

* To transform the conceptual model into a formal or semi-compatible model, you need to **formalize** it using frame-oriented or description logic representation systems.

* Ontologies are built to be reused. In this way, duplication of work in building ontologies has less sense than its duplication when you build a traditional Knowledge Base. So, you should reuse existing ontologies. Try to **integrate** as much as possible existing ontologies in your ontology. A method to integrate ontologically heterogeneous taxonomic knowledge and its application to the medical domain was presented by Gangemi and colleagues in (Gangemi, Steve, & Giacomelli 1996). Farquhar and colleagues (Farquar et al. 1995) have identified four kind of relationships between ontologies that have been integrated: inclusion, polymorphic refinement, circular dependencies and restrictions. Their ontology server provides a semantic model for ontology inclusion in order to avoid conflict between symbols.

* To make your ontology computable, you need to **implement** it in a formal language. As a reference framework for selecting target languages, we would cite the comparative study performed by Speel and colleagues at (Speel et al. 1995) as part of Plinius project (Vet, Speel, & Mars 1995).

* What do you think that it will happen if you use a meta-ontology or ontologies already built with wrong definitions? Probably, your ontology will be wrong too. Before making your ontology available to others, **evaluate** it, that is, make a technical judgment with respect to a frame of reference. A framework for evaluating ontologies was provided by Gómez-Pérez and colleagues at (Gómez-Pérez, Juristo, & Pazos 1995).

* Have you ever try to modify or reuse a program without having a good documentation? Sure, you have. The absence of a sound documentation is also an important obstacle when you reuse/share ontologies already built. So, if you wish your ontology to be reused/shared by others try to **document** it as best you can. No work has been published on this field.

* Anytime, anywhere, someone could ask for including or modifying definitions in the ontology. To **maintain** the ontology is an important activity to be done carefully. Guidelines for maintaining ontologies are also needed.

We remark that the ontology development process does not imply an order on the execution of such tasks. Its goal is only *to identify the list of activities to be done.*

3. Ontology Life Cycle

"Don't build your house starting by the roof", says a Spanish proverb to advice that activities require an order and that they should be divided and carry out step by step in a planned way, in order to succeed. In the previous section, we divided the ontology development process in a set of activities. However, we did not indicate the *order* and *depth* in which such activities should be done. It is obvious that planification and maintenance are the first and the last, but it is not clear whether or not knowledge acquisition is totally or partially simultaneous with the specification and conceptualization activities, if conceptualization precedes to the integration and this one goes before the implementation, if the evaluation and documentation are sequential to the implementation or if they should be done as all the activities move forward, and if you should carry out an activity totally or partially before starting the following.

First, the ontology life cycle answers the previous questions identifying the *set of stages* though which the ontology moves during its life. Making an analogy, we could say that the ontology development process is similar to the production chains in a manufacturing domain as the ontology is to the final product that such production chain

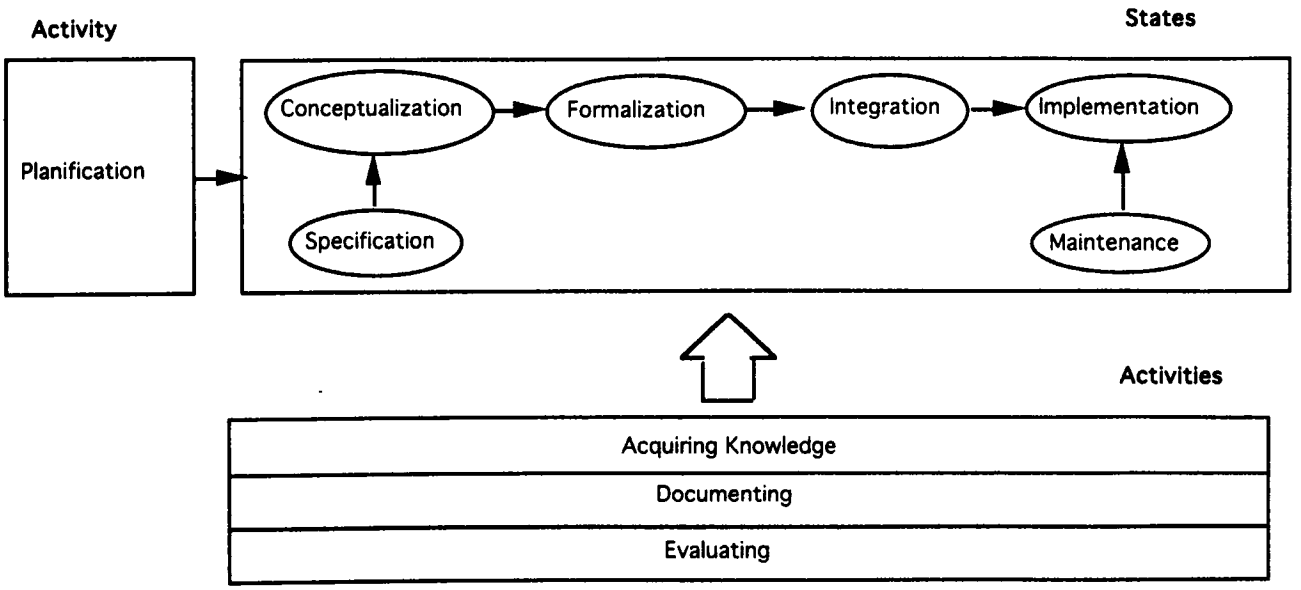

Figure 1. States and activities

creates. Just as the life cycle of human beings moves forward sequentially and irreversibly through the following states: childhood, adolescence, youth, maturity and old age, the ontology life cycle moves forward through the following states: specification, conceptualization, formalization, integration, implementation and maintenance. So, the ontology development process transforms the initial product (the need you have to build the ontology) into a final product (the evaluated, documented ontology, codified in a formal language). The previous states through which the ontology pass conforms its life cycle, as it is shown in figure 1. Knowledge Acquisition, evaluation of ontologies and documentation are tasks that are carried out during the whole life of the ontology as figure 1 shows. In fact, unless the ontologist is an expert in the application domain, most of the acquisition is done simultaneously with the requirements specification phase and decreases as the ontology development process moves forward. In order to prevent error propagation, most of evaluation should be done during the earliest stages of the ontology development process. Finally, you should produce detailed documentation.

Second, the ontology life cycle shows *when* you should perform the activities to move from a given state to the next and in how much *depth*. Some life cycle models have been described in Software Engineering and transferred to Knowledge Engineering (Alonso et al. 1996). As we said before, the main problem facing KE when (s)he begins a new KBS is to build a requirements specification document. Unlike, an ontologist must be able to partially specify a set of requirements before building an ontology, which will constitute the initial core of the ontology. So, the ontology life cycle is closer to a classic software life cycle than it is to a KBS life cycle.

The waterfall life cycle defined by Royce (Royce 1987) is the traditional life cycle model in Software Engineering. In this model, activities are *sequential* in the sense that you cannot move onto the next activity until you have completely finished the previous one. This model forces the ontologist to identify *all* the terms at the beginning, and the implementation must be a mirror of the specification, that is, it must satisfy the complete requirements specification document. Its main inconvenient is that the ontology is static, you cannot include, remove or modify terms in it. Obviously the use of a waterfall life cycle model is not adequate due to the absence of a complete requirements specification at the earliest stages of the development process and due to the evolution of the ontologies definitions over time. Figure 2.a shows how the system grows under this approach. The incremental life cycle (McCracken & Jackson 1982) solves some problems, allowing the partial specification of the requirements. According to this approach, the ontology would grow by layers, allowing the inclusion of new definitions *only* when a new *version* is planned. This model prevents the inclusion of new definitions if they are not planned, but it does permit an incremental development. Figure 2.b shows how the ontology grows according this approach. Finally, the evolving prototype life cycle solves the previous problems since the ontology grows depending on the needs. Indeed, this model lets you to modify, add, and remove definitions in the ontology at any time. So, we think that the evolving prototypes are appropriate life cycle for building ontologies.

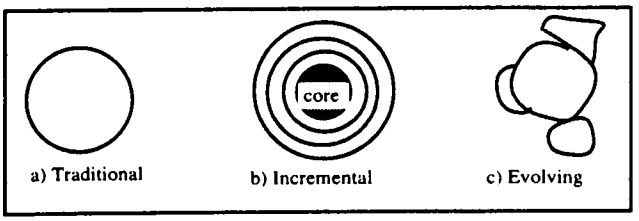

Figure 2. How the ontology grows?

4. METHONTOLOGY: A Methodology to Build Ontologies from Scratch

In general, methodologies give you a set of guidelines of *how* you should carry out the activities identified in the ontology development process, what kinds of techniques are the most appropriate in each activity and what products each one produces. Until now, methodological approaches in building ontologies have been reported by Uschold in the Enterprise ontology, Gruninger in the TOVE project, both in the domain of enterprise modeling (Uschold & Gruninger 1996), and Gómez-Pérez and colleagues in the domain of chemicals (Gómez-Pérez, Fernández, & De Vicente 1996). Figure 3 summarizes the activities proposed and what are equivalent.

Obviously, it is almost impossible to take the three above contributions to propose a general method for building any kind of ontology or meta-ontology. This section presents METHONTOLOGY, a structured method to build ontologies. It is based on the experience acquired in developing an ontology in the domain of chemicals.

4.1 Specification

The goal of the specification phase is to produce either an informal, semi-formal or formal ontology specification document written in natural language, using a set of intermediate representations or using competency questions, respectively. METHONTOLOGY proposes that at least the following information be included:

a) The purpose of the ontology, including its intended uses, scenarios of use, end-users, etc.

b) Level of formality of the implemented ontology, depending on the formality that will be use to codify the terms and their meaning. Uschold (Uschold & Gruninger 1996) classifies the degree or level of formality in a range of: highly informal, semi-informal, semi-formal or rigorously formal ontologies, depending on whether terms and their meanings are codified in a language between natural language and a rigorous formal language.

c) Scope, which includes the set of terms to be represented, its characteristics and granularity.

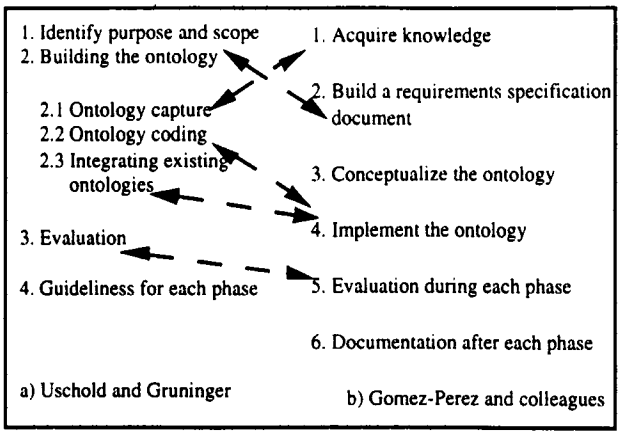

Figure 3.Relationship between phases of two methodologies

The formality of the ontology specification document varies on its degree of formality depending on if you use natural language, competency questions or a middle-out approach. For example, in a middle-out approach, in the scoping activity, you can use a glossary of terms to gather the set of terms that must be included in the ontology, whether or not you know their meaning at this stage of the ontology development process. It is also advisable to group concepts in concepts classifications trees (Gómez-Pérez, Fernández, & De Vicente). The use of these intermediate representations will allow you not only to verify, at the earliest possible stage, relevant terms missed and to include them in the specification document, but also to remove terms that are synonyms and not relevant in your ontology anymore. The goal of these checks is to guarantee the conciseness and completeness of the ontology specification document. Figure 4 shows a short example of an ontology requirements specification document in the domain of chemicals.

Uschold (Uschold & Gruninger 1996) gives an excellent argument on the use of a middle-out as opposed the classic bottom-up and top-down approach in identifying the main terms of your glossary. The main advantage of the middle-out approach is that it allows you to identify the primary concepts of the ontology you are starting on. After reaching agreement on such terms and their definition you can move on to specialize or generalize them, only if they are necessary. As a result, the terms that you use are more stable, and less re-work and overall effort are required.

Since you can not prove the total completeness of your ontology specification document (any time, anywhere, someone may find a new relevant term to be included), you must guarantee in a good ontology specification document must have the following properties:

* Concision, that is, each and every term is relevant, and there are no duplicated or irrelevant terms.

ONTOLOGY REQUIREMENT SPECIFICATION DOCUMENT

Domain: Chemicals
Date: May, 15th 1996
Conceptualized-by: Asunción Gómez-Pérez
Implemented-by: Mariano Fernandez-López

Purpose: Ontology about chemical substances to be used when information about chemical elements is required in teaching, manufacturing, analysis, etc. This ontology coul be used to ascertain, e.g.,the atomic weight of the element Sodium.

Level of Formality: Semi-formal.

Scope: List of 103 elements of substances: *Lithium, Sodium, Chlorine, ...*
List of concepts: *Halogens, noble-gases, semi-metal, metal, ...*
At least information about the following properties: *atomic-number, atomic-weight, atomic-volume-at-20-degrees-celsius, boiling-point, density-at-20-degrees-celsius, electronegativity, electron-affinity, and symbol.*

Sources of Knowledge: *Handbook of Chemistry and Physics*. 65th edition. CRC-Press, Inc. 1984-1985.

Figure 4. Ontology requirements specification in the domain of chemicals.

* Partial completeness, which is related with the coverage of the terms, the stopover problem and level of granularity of each and every term.
* Consistency, which refers to all terms and their meanings making sense in the domain.

4.2. Knowledge Acquisition

It is important to bear in mind that knowledge acquisition is an independent activity in the ontology development process. However, it is coincident with other activities. As we told before, most of the acquisition is done simultaneously with the requirements specification phase, and decreases as the ontology development process moves forward.

Experts, books, handbooks, figures, tables and even other ontologies are sources of knowledge from which the knowledge can be elucidated using in conjunction techniques such us: brainstorming, interviews, formal and informal analysis of texts, and knowledge acquisition tools. For example, if you have no a clear idea of the purpose of your ontology, brainstorming technique, informal interviews with experts, and inspecting similar ontologies will allow you to elaborate a first glossary with terms potentially relevant. To refine the list of terms and their meaning, formal and informal analysis of text techniques in books and handbooks in conjunction with structured and non-structured interviews with experts might be used to include or remove terms in the glossary. Interviews to expert might help you to build concepts classifications trees and to contrast them against figures given in books.

The techniques we used in the knowledge acquisition phase of the chemical ontology were:

* Non-structured interviews with experts, to build a preliminary draft of the requirements specification document.
* Informal text analysis, to study the main concepts given in books and handbooks. This study enables you to fill in the set of intermediate representations of the conceptualization.
* Formal text analysis. The first thing to do is to identify the structures to be detected (definition, affirmation, etc.) and the kind of knowledge contributed by each one (concepts, attributes, values, and relationships).
* Structured interviews with experts to get specific and detailed knowledge about concepts, their properties and their relationships, to evaluate the conceptual model once the conceptualization activity has been finished, and to evaluate implementation.

4.3. Conceptualization

In this activity, you will structure the domain knowledge in a conceptual model that describes the problem and its solution in terms of the domain vocabulary identified in the ontology specification activity. The first thing to do is to build a complete Glossary of Terms (GT). Terms include concepts, instances, verbs and properties. So, the GT identifies and gathers *all* the useful and potentially usable domain knowledge and its meanings. Note that you do not start from scratch when you develop your GT. If you made a good specification document, many terms will have been identified in that document. Other will be identified as the ontology construction process advances. Then, these new terms must be included in the GT.

Once you have almost completed the GT, you must group terms as concepts and verbs. Each set of concepts/verbs would include concepts/verbs that are closely related to other concepts/verbs inside the same group as opposed to other groups. Indeed, for each set of related concepts and related verbs, a concepts classification tree and a verbs diagram is built. After they have been built, you can split your ontology development process into different, but related, teams. Those related with concepts, should follow the guidelines presented by Gómez-Pérez and colleagues in (Gómez-Pérez, Fernández, & De Vicente 1996), and those encharged of conceptualizing verbs are presented at (Vicente 1997). Figure 5 graphically summarizes the intermediate representations used in the conceptualization phase.

Concepts will be described using (Gómez-Pérez, Fernández, & De Vicente 1996): *Data Dictionary*, which describes and gathers all the useful and potentially usable domain concepts, their meanings, attributes, instances, etc.; *tables of instance attributes*, which provide information about the attribute or about its values at the instance; *tables*

of class attributes, to describe the concept itself, not its instances; *tables of constants*, used to specify information related to the domain of knowledge that always take the same value; *tables of instances*, which define instances; and *attributes classification trees*, to graphically display attributes and constants related in the inference sequence of the root attributes, as well as the sequence of formulas or rules to be executed to infer such attributes.

Verbs represent actions in the domain. They could be described using (Vicente 1997): a *Verbs Dictionary*, to express the meaning of verbs in a declarative way; *tables of conditions*, which specify a set of conditions to be satisfied before executing an action, or a set of conditions to be guaranteed after the execution of an action. Finally, to say that *tables of formulas* and *tables of rules* gather knowledge about formulas and rules. Note that both branches use these two intermediate representations.

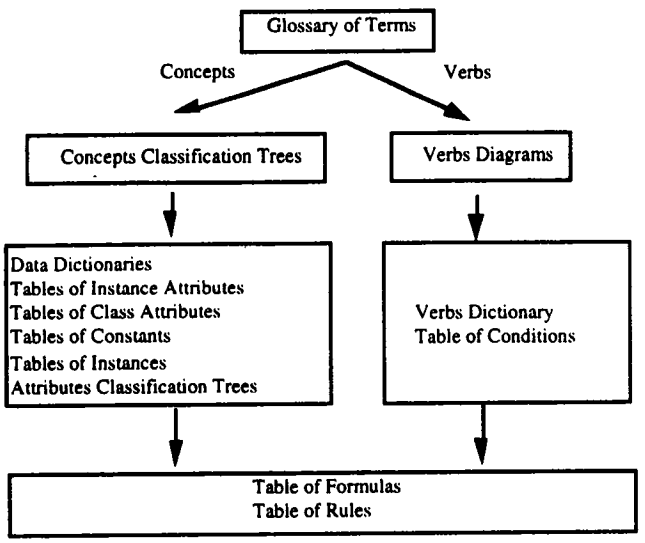

Figure 5. Set of Intermediate Representations in the conceptualization phase.

In sum, METHONTOLOGY produces in this phase a conceptual model expressed as a set of well-defined deliverables that will allow to the final users: (a) to ascertain whether or not an ontology is useful and usable for a given application without inspecting its source code; and (b) to compare the scope and completeness of several ontologies, their reusability, and shareability by analyzing the knowledge expressed in each IR.

4.4. Integration

With the goal of speeding up the construction of your ontology, you might consider reuse of definitions already built into other ontologies instead of starting from scratch. In this case, we propose the following:

1. Inspect meta-ontologies (i.e., in Cyc, in Ontolingua, ...) to select those that better fit your conceptualization. The goal is to guarantee that the sets of new and reused definitions are based upon the same set of basic terms. If existing meta-ontologies are not appropriate for your ontology, you should start the definition and implementation of a new meta-ontology in a formal language.
2. Whether or not you reuse existing meta-ontologies, the next step is to find out which libraries of ontologies provide definitions of terms whose semantic and implementation is coherent with the terms identified in your conceptualization. Once you have chosen the most appropriate terms, if the meta-ontology upon which those terms have been built is different of the meta-ontology used to build the yours, you should check the existence of translators to transform definitions into your target language.

Sometimes it might occur that a term in your conceptualization (e.g., centimeter), that should be included in a given ontology (e.g., Standard Units in Ontolingua) is not provided by the ontology server. In this case, you should justify the need to include the missed definitions as well as the benefits of such inclusion to the ontology maintainer.

As a result of this activity, METHONTOLOGY proposes the development of an integration document, summarizing: the meta-ontology you will use and, for each and every term whose definition is going to be used: the name of the term in the conceptual model, the name of the ontology from which you will take its definition, the name of the definition and its arguments in the ontology, as shown in figure 6.

Meta-Ontology	The frame-ontology in Ontolingua	
Term in your Conceptualization	**Ontology to be reused**	**Name of the term in the ontology**
Kilometer	Standard-Units in Ontolingua	Kilometer
Centimeter	Standard-Units in Ontolingua	Undefined
Exponent	KIF-Numbers in Ontolingua	Expt

Figure 6. An example of an integration document

4.5. Implementation

Ontologies implementation requires the use of an environment that supports the meta-ontology and ontologies selected at the integration phase. The result of this phase is the ontology codified in a formal language such us: CLASSIC, BACK, LOOM, Ontolingua, Prolog, C++ or in your favorite language.

Any ontology development environment should provide, at least: a lexical and syntactic analyzer to guarantee the absence of lexical and syntactic errors; translators, to guarantee the portability of the definitions into other target languages; an editor, to add, remove or modify definitions; a browser, to inspect the library of

ontologies and their definitions; a searcher, to look for the most appropriate definitions; evaluators, to detect incompleteness, inconsistencies and redundant knowledge; an automatic maintainer, to manage the inclusion, removal or modification of existing definitions, and so on.

4.6. Evaluation

A framework for evaluating knowledge sharing technology (software, ontologies and documentation) has been presented by Gómez Pérez and colleagues in (Gómez-Pérez, Juristo, & Pazos 1995). *Evaluation* means to carry out a technical judgment of the ontologies, their software environment and documentation with respect to a frame of reference (in our case the requirements specification document) during each phase and between phases of their life cycle. Evaluation subsumes the terms Verification and Validation. *Verification* refers to the technical process that guarantees the *correctness* of an ontology, its associated software environments, and documentation with respect to a frame of reference during each phase and between phases of their life cycle. *Validation* guarantees that the ontologies, the software environment and documentation correspond to the system that they are supposed to represent. Based on the experience of verifying Ontolingua ontologies, a set of guidelines and how to look for incompleteness, inconsistencies and redundancies have been presented in (Gómez-Pérez 1997).

The output proposed by METHONTOLOGY for this activity is many evaluation document in which the ontologist will describe how the ontology has been evaluated, the techniques used, the kind of errors found in each activity, and the sources of knowledge used in the evaluation.

4.7. Documentation

There are not consensuated guidelines on how to document ontologies. In many cases, the only documentation available is in the code of the ontology, the natural language text attached to formal definitions, and papers published in conference proceedings and journals settings out important questions of the ontology already build. This problem is the result of a vicious circle: almost anyone documents ontologies due to there are no guidelines to perform it, there are no guidelines to document ontologies because of the absence of methodologies to build ontologies, and there are no standard methodologies to build ontologies due to ontologist do not write, during the whole ontology development process, the steps they follow to build ontologies.

METHONTOLOGY pretends to break this circle including the documentation as an activity to be done during the whole ontology development process. In fact, after the specification phase, you get a *requirements specification document*; after the knowledge acquisition phase, a *knowledge acquisition document*; after the conceptualization, a *conceptual model document* that includes a set of intermediate representations that describe the application domain; after the formalization, a *formalization document*; after the integration, an *integration document*; after the implementation, the *implementation document*; and during the evaluation, an evaluation *document*.

5. Conclusion

In this paper we have reduced the existing gap between ontological art and ontological engineering by:

1. Identifying a set of *activities* to be done during the ontology development process. They are: planify, specify, acquire knowledge, conceptualize, formalize, integrate, implement, evaluate, document, and maintain.
2. Proposing the *evolving prototype* as the life cycle that better fits with the ontology life cycle. The life of an ontology moves on through the following *states*: specification, conceptualization, formalization, integration, implementation, and maintenance. The evolving prototype life cycle allows the ontologist to go back from any state to other if some definition is missed or wrong. So, this life cycle permits the inclusion, removal or modification of definitions anytime of the ontology life cycle. Knowledge acquisition, documentation and evaluation are support activities that are carried out during the majority of these states.
3. Defining METHONTOLOGY, a well structured methodology to build ontologies from scratch. The methodologies includes a set of activities, techniques to carry out each one, and deliverables to be produced after the execution of such activities using its attached techniques. METHONTOLOGY highly recommend the reuse of existing ontologies

References

Alonso, F.; Juristo, N.; Maté, J.L.; Pazos, J. *Software Engineering and Knowledge Engineering: Towards a Common Life-Cycle.* **Journal of Systems and Software**. N° 33. 1996. Pags 65-79.

Farquhar, A. Fikes, R.; Pratt, W.; Rice, J. *Collaborative Ontology Construction for Information Integration.* KSL-95-63. Technical Report Knowledge Systems Laboratory. Stanford University. Ca. August 1995.

Gangemi, A.; Steve, G.; Giacomelli, F; *ONIONS: an ontological methodology for taxonomic knowledge integration.* **Working notes of the workshop Ontological Engineering. ECAI'96.** Pags. 29-40.

Gómez-Pérez, A.; Juristo, N.; Pazos, J. *Evaluation and Assessment of Knowledge Sharing Technology.* **Towards**

Very Large Knowledge Bases. Ed. by N. Mars. IOS Press. Amsterdam. 1995. Pags. 289-296.

Gómez-Pérez, A.; Fernández, M; De Vicente, A.; *Towards a Method to Conceptualize Domain Ontologies.* **Working notes of the wokshop Ontological Engineering. ECAI'96.** Pags. 41-52.

Gómez-Pérez, A. *A framework to Verify Knowledge Sharing Technology.* **Expert Systems with Application.** To be published

McCracken; M. A. Jackson. *Life Cycle Concept Considered Harmful.* **ACM Software Engineering notes.** April 1982. Pags 29-32.

Royce W. M. *Managing the Development of Large Software Systems.* **Proc. 9th International Conference Software Engineering. IEEE.** Computer Society. 1987. Pags. 328-338

Speel, H.; Raalte, F; Vet, P.; Mars, N. *Scalability of the Performance of Knowledge Represenation Systems.* **Towards Very Large Knowledge Bases.** Ed. by N. Mars. IOS Press. Amsterdam. 1995. Pags. 173-184.

Uschold, M.; Gruninger, M. *ONTOLOGIES: Principles, Methods and Applications.* **Knowledge Engineering Review.** Vol. 11; N. 2; June 1996.

Vet, P.; Speel, P.; Mars, N. *Ontologies for Very Large Knowledge Bases in Material Science: a Case Study.* **Towards Very Large Knowledge Bases.** Ed. by N. Mars. IOS Press. Amsterdam. 1995. Pags. 73-83.

Vicente, A; **Concepualizacion de verbos en ontologías de dominio.** Tésis de Master en Ingeniería del Conocimiento. Facultad de Informatica. Universidad Politécnica de Madrid. 1997. To be published.