Building legal ontologies with METHONTOLOGY and WebODE

Oscar Corcho[1], Mariano Fernández-López, Asunción Gómez-Pérez, Angel López-Cima

Facultad de Informática. Universidad Politécnica de Madrid
Campus de Montegancedo, s/n. 28660 Boadilla del Monte. Madrid. Spain
{ocorcho,mfernandez,asun,alopez}@fi.upm.es

Abstract. This paper presents how to build an ontology in the legal domain following the ontology development methodology METHONTOLOGY and using the ontology engineering workbench WebODE. Both of them have been widely used to develop ontologies in many other domains. The ontology used to illustrate this paper has been extracted from an existing class taxonomy proposed by Breuker, and adapted to the Spanish legal domain.

1. Introduction

When the application of the technology in a specific area attains some degree of maturity, it stops being an art and becomes an engineering. A characteristic of an engineering is that it provides methods, methodologies and tools to perform the tasks required in such area. Methodologies state “what”, “who” and “when” a given activity should be performed [7], and tools give support to such activities. Ontological Engineering refers to the set of activities that concern the ontology development process, the ontology life cycle, the methods and methodologies for building ontologies, and the tool suites and languages that support them [12].

Some outstanding works on how to develop ontologies methodologically are the following: Uschold and King’s [24], Grüninger and Fox’s [14], METHONTOLOGY [9, 10], and On-To-Knowledge [20], among others. Concerning software platforms that aid in ontology development, we can mention Protégé-2000 [18], OntoEdit [21], KAON [17], and WebODE [1], among others.

In this paper we present how to develop a legal entity ontology following METHONTOLOGY and using WebODE (the methodology and software platform proposed by the Ontological Engineering Group at UPM). With them we have built ontologies in different domains, like Chemistry, Science, knowledge management, e-commerce, etc.

This paper is addressed to experts in Law who want to build ontologies in that domain. We present how we have adapted a class taxonomy proposed by Breuker[2], to

[1] Now at Intelligent Software Components (ocorcho@isoco.com)
[2]http://zeus.ics.forth.gr/forth/ics/isl/projects/ontoweb/_notes/legal-ontol-ontoweb-sard-2002.ppt

build a legal entity ontology in the context of the Spanish legal domain. We have used METHONTOLOGY (section 2) and WebODE (section 3) as our methodological and technological frameworks. Section 4 describes briefly other methods and methodologies, and tools. Finally, section 5 presents conclusions to this work.

2. Building a legal entity ontology according to METHONTOLOGY

2.1 METHONTOLOGY in a nutshell

METHONTOLOGY [9, 10] was developed within the Ontological Engineering group at Universidad Politécnica de Madrid. This methodology enables the construction of ontologies at the knowledge level, and has its roots in the main activities identified by the IEEE software development process [15] and in other knowledge engineering methodologies [13].

ODE and WebODE [1] were built to give technological support to METHONTOLOGY. Other ontology tools and tool suites can also be used to build ontologies following this methodology, for example the ones mentioned in the introduction: Protégé-2000 [18], OntoEdit [21], KAON [17], etc. METHONTOLOGY has been proposed[3] for ontology construction by the Foundation for Intelligent Physical Agents (FIPA), which promotes inter-operability across agent-based applications.METHONTOLOGY guides in how to carry out the whole ontology development through the specification, the conceptualization, the formalization, the implementation and the maintenance of the ontology (see figure 1). We now describe briefly what each activity consists in:

- The *specification* activity states why the ontology is being built, what its intended uses are and who the end-users are.
- The *conceptualization* activity in METHONTOLOGY organizes and converts an informally perceived view of a domain into a semi-formal specification using a set of intermediate representations (IRs) based on tabular and graph notations that can be understood by domain experts and ontology developers. The result of the conceptualization activity is the ontology conceptual model. The *formalization* activity transforms the conceptual model into a formal or semi-computable model. The *implementation* activity builds computable models in an ontology language (Ontolingua [8], RDF Schema [4], OWL [5], etc.). Tools implement automatically conceptual models in varied ontology languages. For example, WebODE imports and exports ontologies from and to the following languages: XML, RDF(S), OIL, DAML+OIL, OWL, CARIN, FLogic, Jess, and Prolog.
- The *maintenance* activity updates and corrects the ontology if needed.

METHONTOLOGY also identifies management activities (schedule, control, and quality assurance), and support activities (knowledge acquisition, integration, evaluation, documentation, and configuration management).

[3] http://www.fipa.org/specs/fipa00086/

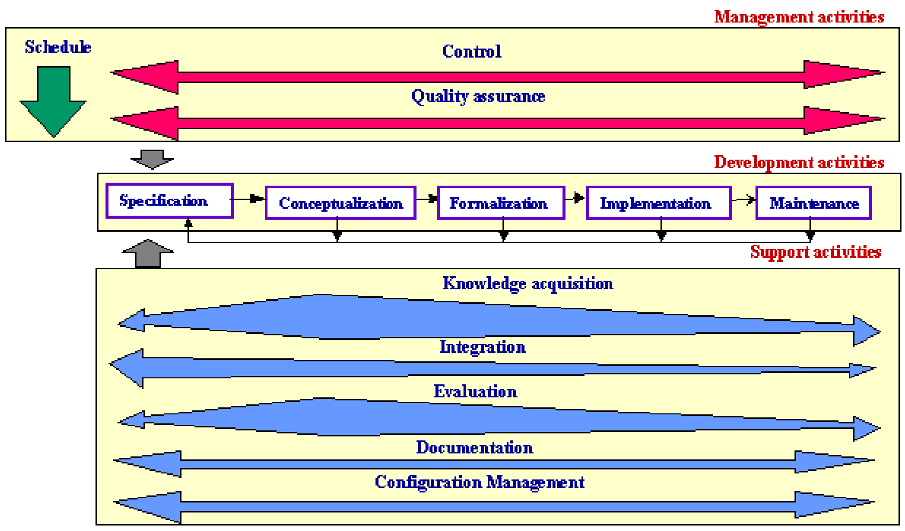


Fig. 1: Activities in the ontology development proposed by METHONTOLOGY.

In the next sections we show the process followed to conceptualize an ontology about legal entities (juridical persons, organizations, etc.) in the Spanish legal domain. As already commented above, we have adapted a class taxonomy of legal entities proposed by Breuker for the Spanish legal domain. The definitions provided for some of the legal terms of this ontology are adapted from the LEGAMedia lexicon[4].

2.2 Main ontology modelling components

METHONTOLOGY proposes to conceptualize ontologies with a set of tabular and graphical IRs. Such IRs allow modeling the components described in this section.

Concepts are taken in a broad sense. For instance, in the legal domain, concepts are: `juridical person`, `court`, `juvenile`, etc. Concepts in the ontology are usually organized in taxonomies through which inheritance mechanisms can be applied. For instance, we can represent a taxonomy of legal entities (which distinguishes persons and organizations), where a `juridical person` is a subclass of a `person`, a `company` is a subclass of a `juridical person`, a `private company` is a subclass of `company`, etc.

Relations represent a type of association between concepts of the domain. If the relation links two concepts, for example, `hears`, which links `court` to `lawsuit`, it is called binary relation. An important binary relation is *Subclass-Of*, which is used for building the class taxonomy, as shown above. Each binary relation may have an inverse relation that links the concepts in the opposite direction. For example, the relation `is heard` is the inverse of `hears`.

[4] http://www.legamedia.net/lx/lx.php

Instances are used to represent elements or individuals in an ontology. An example of instance of the concept `Court` is `Albacete Provincial Court` or `Constitutional Court`. Relations can be also instantiated. For example, we can express that Albacete Provicial Court hears 127/2004 lawsuit as follows: `hears (Albacete Provincial Court, 127/2004 lawsuit)`, using a first order logic notation.

Constants are numeric values that do not change during much time. For example, `adult age in Spain`.

Attributes describe properties of instances and of concepts. We can distinguish two types of attributes: instance and class attributes. ***Instance attributes*** describe concept instances, where they take their values. These attributes are defined in a concept and inherited by its subconcepts and instances. For example, the `first name` of a `physical person` is proper to each instance. ***Class attributes*** describe concepts and take their values in the concept where they are defined. Class attributes are neither inherited by the subclasses nor by the instances. An example is the attribute `type of control` of the concept `company`, which can be used to determine the type of control of a `private company`, a `public company`, and a `shared control company`. Ontology development tools usually provide predefined domain-independent class attributes for all the concepts, such as the concept documentation, synonyms, acronyms, etc. Besides, other user-defined domain-dependent class attributes can be usually created.

Formal axioms are logical expressions that are always true and are normally used to specify constraints in the ontology. An example of axiom is that a person cannot be the defendant and the plaintiff in the same lawsuit.

Rules are generally used to infer knowledge in the ontology, such as attribute values, relation instances, etc. An example of rule is that lawsuits where juveniles up 14 years old are the defendants are heard by a juvenile court.

2.3 Conceptualization of a legal entity ontology

When building ontologies, ontologists should not be anarchic in the use of the above modeling components during the ontology conceptualization. They should not define, for instance, a relation if the linked concepts are not precisely defined in the ontology. METHONTOLOGY includes in the conceptualization activity the set of structuring knowledge tasks shown in figure 2.

The figure emphasizes the ontology components (concepts, attributes, relations, constants, formal axioms, rules, and instances) built inside each task, and illustrates the order proposed to create such components during the conceptualization activity. This modeling process is not sequential, though some order must be followed to ensure the consistency and completeness of the knowledge represented. If new vocabulary is introduced, the ontologist can return to any previous task.

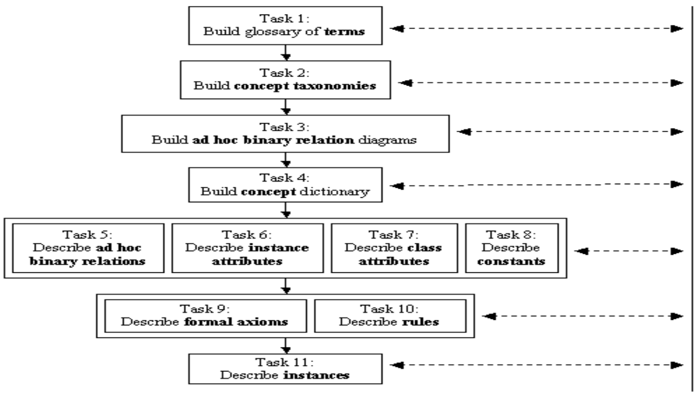

Task 1: Build glossary of terms
Task 2: Build concept taxonomies
Task 3: Build ad hoc binary relation diagrams
Task 4: Build concept dictionary
Task 5: Describe ad hoc binary relations
Task 6: Describe instance attributes
Task 7: Describe class attributes
Task 8: Describe constants
Task 9: Describe formal axioms
Task 10: Describe rules
Task 11: Describe instances

Fig. 2: Tasks of the conceptualization activity according to METHONTOLOGY.

Task 1: To build the glossary of terms. First, the ontologist builds a glossary of terms that includes all the relevant terms of the domain (concepts, instances, attributes, relations between concepts, etc.), their natural language descriptions, and their synonyms and acronyms. Table 1 illustrates a section of the glossary of terms of the legal entity ontology. It is important to mention that on the initial stages of the ontology conceptualization the glossary of terms might contain several terms that refer to the same compoment. Then the ontologist should detect that they appear as synonyms.

Name	Synonyms	Acronyms	Description	Type
adult age in Spain	--	--	The adult age in Spain is 18	Constant
court	juridical tribunal	--	Although 'court' can be understood as a physical place or as a judge, we assume (in this ontology) that a court is a judicial tribunal	Concept
birth day	--	--	The day when a person was born	Instance Attribute
is defendant*(person, lawsuit)*	--	--	It is the lawsuit of a defendant	Relation

Table 1: An excerpt of the Glossary of Terms of the legal entity ontology.

Task 2: To build concept taxonomies. When the glossary of terms contains a sizable number of terms, the ontologist builds concept taxonomies to define the concept hierarchy.

To build concept taxonomies, the ontologist selects terms that are concepts from the glossary of terms. METHONTOLOGY proposes to use the four taxonomic relations defined in the Frame Ontology [8] and the OKBC Ontology [5]: *Subclass-Of*, *Disjoint-Decomposition*, *Exhaustive-Decomposition*, and *Partition*.

A concept C_1 is a *Subclass-Of* another concept C_2 if and only if every instance of C_1 is also an instance of C_2. For example, as Fig. 3 illustrates, `physical person` is a subclass of `person`, since every physical person is a person. A concept can be a subclass of more than one concept in the taxonomy. For instance, the concept `shared control company` is a subclass of the concepts `private company` and `public company`, since a shared control company is controlled by private and public entities.

A *Disjoint-Decomposition* of a concept C is a set of subclasses of C that do not have common instances and do not cover C, that is, there can be instances of the concept C that are not instances of any of the concepts in the decomposition. For example (see Fig. 3), the concepts `ministry` and `court` make up a disjoint decomposition of the concept `organization` because no organization can be simultaneously a ministry, and a court. Besides, there may be instances of the concept `organization` that are not instances of any of the two classes.

An *Exhaustive-Decomposition* of a concept C is a set of subclasses of C that cover C and may have common instances and subclasses, that is, there cannot be instances of the concept C that are not instances of at least one of the concepts in the decomposition. For example (see Fig. 3), the concepts `private company` and `public company` make up an exhaustive decomposition of the concept `company` because there are no companies that are not instances of at least one of those concepts, and those concepts can have common instances. For example, a `shared control company` is a `public company` and a `private company`.

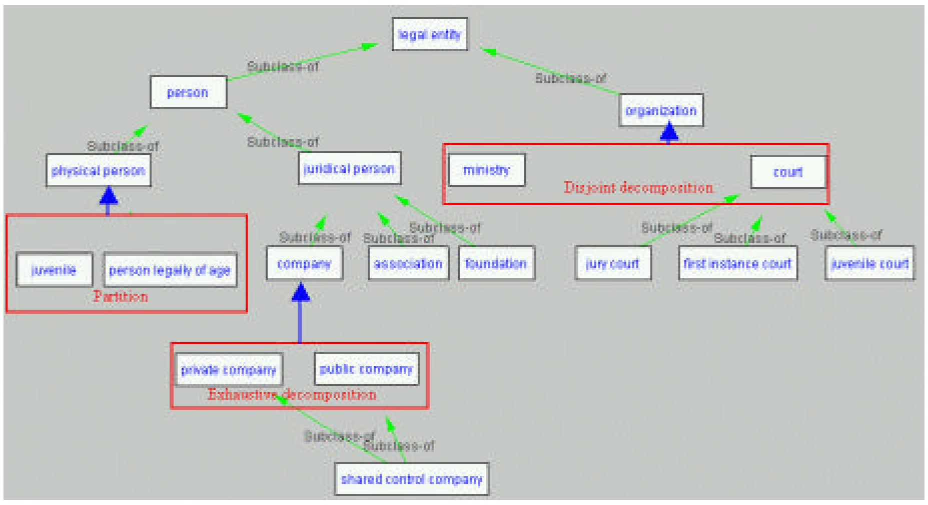


Fig. 3: An excerpt of the Concept Taxonomy of the legal entity ontology.

A *Partition* of a concept C is a set of subclasses of C that do not share common instances and that cover C, that is, there are not instances of C that are not instances of one of the concepts in the partition. For example, Fig. 3 shows that the concepts `juvenile` and `person legally of age` make up a partition of the concept `physical person` because every physical is either juvenile or person legally of age.

Once the ontologist has structured the concepts in the concept taxonomy, and before going ahead with the specification of new knowledge, s(he) should examine that the taxonomies contain no errors [11]. For example, it should be checked that an element is not simultaneously instance of two classes of a disjoint decomposition, that there are not loops in the concept taxonomy, that several terms do not refer to the same concept, etc.

Task 3: To build ad hoc binary relation diagrams. Once the taxonomy has been built and evaluated, the conceptualization activity proposes to build ad hoc binary relation diagrams. The goal of this diagram is to establish ad hoc relationships between concepts of the same (or different) concept taxonomy. Figure 4 presents a fragment of the ad hoc binary relation diagram of our legal entity ontology, with the relations `is plaintiff`, `is defendant` and `hears`, and their inverses `has plaintiff`, `has defendant` and `is heard`. Such relations connect the root concepts (`person` and `lawsuit`, and `court` and `lawsuit`) of the concept taxonomies of legal entities and lawsuits. From an ontology integration perspective, such ad hoc relations express that the legal entity ontology will include the lawsuit ontology and vice versa.

Before going ahead with the specification of new knowledge, the ontologist should check that the ad hoc binary diagrams have no errors. The ontologist should figure out whether the domains and ranges of each argument of each relation delimit exactly and precisely the classes that are appropriate for the relation. Errors appear when the domains and ranges are imprecise or over-specified.

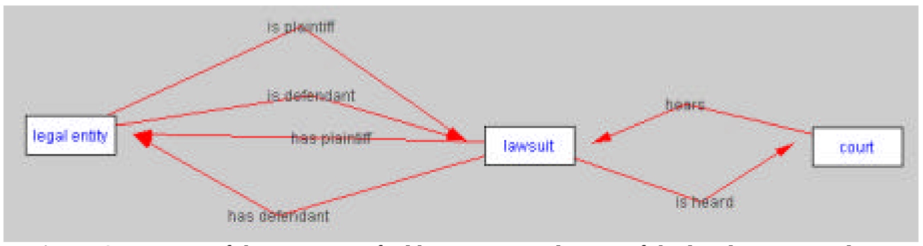


Fig. 4: An excerpt of the Diagram of ad hoc Binary Relations of the legal entity ontology.

Task 4: To build the concept dictionary. Once the concept taxonomies and ad hoc binary relation diagrams have been generated, the ontologist must specify which are the properties and relations that describe each concept of the taxonomy in a concept dictionary, and, optionally, their instances.

A concept dictionary contains all the domain concepts, their relations, their instances, and their class and instance attributes. The relations specified for each concept are those whose domain is the concept. For example, the concept `person` has two relations: `is plaintiff` and `is defendant`. Relations, instance attributes and class attributes are local to concepts, which means that their names can be repeated in different concepts. Table 2 shows a small section of the concept dictionary of the legal entity ontology.

Concept name	Instances	Class attributes	Instance attributes	Relations
court	Constitutional Court National Court Supreme Court Albacete Provincial Court	--	number of members seat territorial jurisdiction	hears
company	--	type of control	name	--
lawsuit	--	--	--	has defendant has plaintiff is heard
person	--	--	--	is defendant is plaintiff
physical person	--	--	age birth day death day first family name first name nationality second family name	is mother of has father has mother is father of

Table 2: An excerpt of the Concept Dictionary of the legal entity ontology.

As we said before, once the concept dictionary has been built, the ontologist must describe in detail each of the ad hoc binary relations, class attributes, and instance attributes appearing in it. In addition, the ontologist must describe accurately each of the constants that appear in the glossary of terms. Though METHONTOLOGY does all these tasks, it does not propose a specific order to perform them.

Task 5: To define ad hoc binary relations in detail. The goal of this task is to describe in detail all the ad hoc binary relations included in the concept dictionary, and to produce the ad hoc binary relation table. For each ad hoc binary relation, the ontologist must specify its name, the names of the source and target concepts, its cardinality, and its inverse relation. Table 3 shows a section of the ad hoc binary relation table of the legal entity ontology, which contains the definition of the relations `is defendant`, `is plaintiff`, etc.

Relation name	Source concept	Source cardinality (Max)	Target concept	Inverse relation
is defendant	Person	N	lawsuit	has defendant
is plaintiff	Person	N	lawsuit	has plaintiff
hears	Court	N	lawsuit	is heard
has defendant	Lawsuit	N	person	is defendant
has plaintiff	Lawsuit	N	person	is plaintiff
is heard	Lawsuit	N	court	hears

Table 3: An excerpt of the ad hoc Binary Relation Table of the legal entity ontology.

Task 6: To define instance attributes in detail. The aim of this task is to describe in detail all the instance attributes already included in the concept dictionary by means of an instance attribute table. Each row of the instance attribute table contains the detailed description of an instance attribute. Instance attributes are those attributes that describe the instances of the concept and whose value(s) may be different for each instance of the concept. For each instance attribute, the ontologist must specify the following fields: its name; the concept it belongs to (attributes are local to concepts); its value type; and range of values (in the case of numerical values); minimum and maximum cardinality; instance attributes, class attributes and constants used to infer values of the attribute; attributes that can be inferred using values of this attribute; formulae or rules that allow inferring values of the attribute; and references used to define the attribute. Table 4 shows a fragment of the instance attribute table of the legal entity ontology. Some of the previous fields are not shown for the sake of space. This table contains some of the instance attributes of the concept `court`: `number of members`, `seat`, and `territorial jurisdiction`.

The use of measurement units in numerical attributes causes the integration of the *Standard Units* ontology. This is an example of how METHONTOLOGY proposes to integrate ontologies during the conceptualization activity, and not to postpone the integration to the ontology implementation activity.

Instance attribute name	Concept name	Value type	Value Range	Cardinality
number of members	court	Integer	1 ..	(1, 1)
seat	court	String	--	(1, 1)
territorial jurisdiction	court	String	--	(1, 1)

Table 4: An excerpt of the Instance Attribute Table of the legal entity ontology.

Task 7: To define class attributes in detail. The aim of this task is to describe in detail all the class attributes already included in the concept dictionary by means of a class attribute table. Each row of the class attribute table contains a detailed description of the class attribute. For each class attribute, the ontologist should fill the following information: name; the name of the concept where the attribute is defined; value type; value(s); cardinality; the instance attributes whose values can be inferred with the value of this class attribute; etc. For example, the class attribute type of control would be defined for the concepts private company and public company as presented in table 5.

Class attribute name	Defined concept	Value type	Cardinality	Values
type of control	private company	[private,public]	(1,2)	private
type of control	public company	[private,public]	(1,2)	public

Table 5: An excerpt of the Class Attribute Table of the legal entity ontology.

Task 8: To define constants in detail. The aim of this task is to describe in detail each of the constants defined in the glossary of terms. Each row of the constant table contains a detailed description of a constant. For each constant, the ontologist must specify the following: name, value type (a number, a mass, etc.), value, the measurement unit for numerical constants, and the attributes that can be inferred using the constant. Table 6 shows a fragment of the constant table of our legal entity ontology, where the constant `adult age in Spain` is defined. The attributes that can be inferred with the constant are omitted.

Name	Value type	Value	Measurement unit
adult age in Spain	Cardinal	18	year

Table 6: An excerpt of the Constant Table of the legal entity ontology.

METHONTOLOGY proposes to describe formal axioms and rules in parallel once concepts and their taxonomies, ad hoc relations, attributes, and constants have been defined.

Task 9: To define formal axioms. To perform this task, the ontologist must identify the formal axioms needed in the ontology and describe them precisely. For each formal axiom definition, METHONTOLOGY proposes to specify the following information: name, NL description, the logical expression that formally describes the axiom using first order logic, the concepts, attributes and ad hoc relations to which the axiom refers, and the variables used.

Axiom name	Description	Expression	Referred concepts	Referred relations	Variables
incompatibility plaintiff defendant	A person cannot be plaintiff and defendant in the same lawsuit	not (exists(?X,?Y) (person(?X) and lawsuit(?Y) and [is plaintiff](?X,?Y) and [is defendant](?X,?Y)))	person lawsuit	is plaintiff is defendant	?X ?Y

Table 7: An excerpt of the Formal Axiom Table of the lawsuit ontology.

As we have already commented, METHONTOLOGY proposes to express formal axioms in first order logic. Table 7 shows a formal axiom in our legal entity ontology that states that “A person cannot be plaintiff and defendant in the same lawsuit”. The columns that correspond to the referred concepts and relations contain the concepts and relations that are used inside the formal axiom. The variables used are ?X for `person`, and ?Y for the `lawsuit`.

We must note that the definition of the logical expression may be difficult for an expert with no experience in first order logic.

Task 10: To define rules. Similarly to the previous task, the ontologist must identify first which rules are needed in the ontology, and then describe them in the rule table. For each rule definition, METHONTOLOGY proposes to include the following information: name, NL description, the expression that formally describes the rule, the concepts, attributes and relations to which the rule refers, and the variables used in the expression.

METHONTOLOGY proposes to specify rule expressions using the template *if <conditions> then <consequent>*. The left-hand side of the rule consists of conjunctions of atoms, while the right-hand side of the rule is a single atom.

Table 8 shows a rule that states and establishes that "Lawsuits where juveniles up 14 years old are defendants are heard by a juvenile court". This rule would let us infer the type of court for juveniles. As shown in the table, the rule refers to the concepts `juvenile`, `lawsuit` and `court`, to the attribute `age`, and to the relations `is defendant` and `hears`. The variables used are ?X for the `juvenile`, ?Y for the `integer`, `lawsuit` for `?Z` and ?Z for `court`.

As in the case of formal axioms, the definition of the rule expression may be difficult for experts who have little experience in first order logic.

Rule name	Description	Expression	Concepts	Referred attributes	Referred relations	Variables
juvenile courts for juveniles	Lawsuits where juveniles up 14 years old are defendants are heard by a juvenile court	If juvenile(?X) and lawsuit(?Z) and court(?W) and age(?X, ?Y) and ?Y > 14 and [is defendant](?X, ?Z) and hears(?W, ?Z) then [juvenile court](?W)]	juvenile lawsuit court	age	is defendant hears	?X ?Z ?W

Table 8: An excerpt of the Rule Table of the legal entity ontology.

Task 11: To define instances. Once the conceptual model of the ontology has been created the ontologist might define relevant instances that appear in the concept dictionary inside an instance table. For each instance, the ontologist should define: its name, the name of the concept it belongs to, and its attribute values, if known. Table 9 presents some instances of the instance table of our legal entity ontology: `National Court`, `Supreme Court` and `Constitutional Court`). All of them are instances of the concept `court`, as defined in the concept dictionary, and they have some attribute and relation values specified, for: `seat`, `territorial jurisdiction`, and `number of members`. These instances could have more than one value for the attributes whose maximum cardinality is higher than one.

Instance name	Concept name	Attribute	Values
National Court	court	seat	Madrid
		territorial jurisdiction	Spain
Supreme Court	court	territorial jurisdiction	Spain
Constitutional Court	court	number of members	12
		territorial jurisdiction	Spain

Table 9: An excerpt of the Instance Table of the legal entity ontology.

METHONTOLOGY has been used by different groups to build ontologies on Chemistry, Science, knowledge management, e-commerce, etc. A detailed description of this ontology building methodology can be found in [12].

3. Building a legal ontology with WebODE

WebODE[5] [1] is an ontological engineering workbench developed by the Ontological Engineering group at Universidad Politécnica de Madrid (UPM). The current version is 2.0. WebODE is the offspring of the ontology design environment ODE, a standalone ontology tool based on tables and graphs, which allowed users to customize the knowledge model used for conceptualizing their ontologies according to their KR needs. Both ODE and WebODE give support to the ontology building methodology METHONTOLOGY, described in the previous section.

Currently, WebODE contains an ontology editor, which integrates most of the ontology services offered by the workbench, an ontology-based knowledge management system (ODEKM), an automatic Semantic Web portal generator (ODESeW), a Web resources annotation tool (ODEAnnotate), and a Semantic Web services editing tool (ODESWS). A detailed description of all of them can be found in [12].

Let us start describing the WebODE ontology editor. The editor is a Web application built on top of the ontology access service (ODE API), which integrates several ontology building services from the workbench: ontology edition, navigation, documentation, merge, reasoning, etc.

Three user interfaces are combined in this ontology editor: an HTML form-based editor for editing all ontology terms except axioms and rules; a graphical user interface, called ODEDesigner, for editing concept taxonomies and relations graphically; and WAB (WebODE Axiom Builder), for editing formal axioms and rules. We now describe them and highlight their most important features.

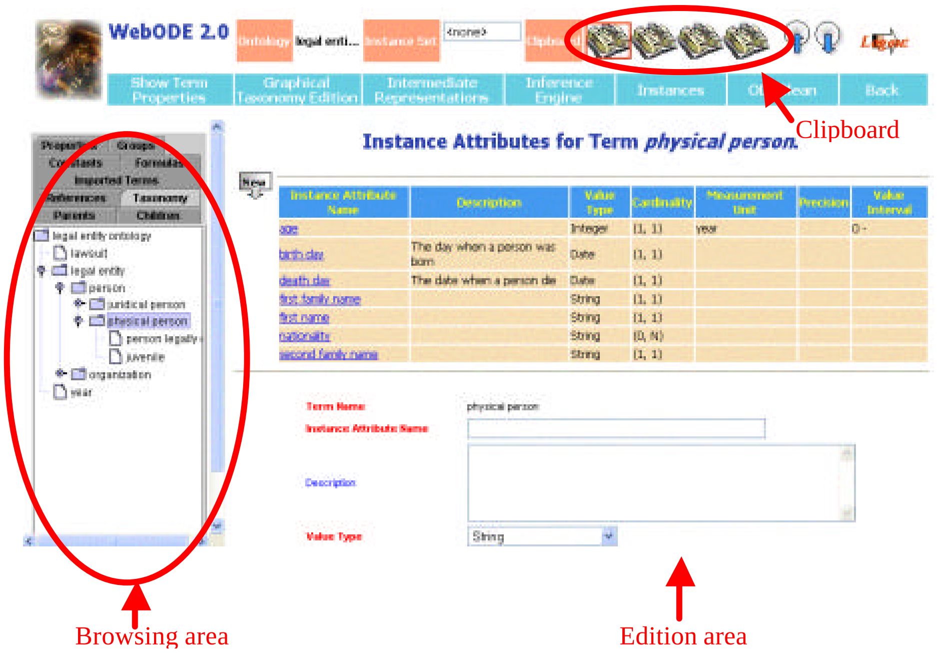
Figure 5 shows a screenshot of the HTML interface for editing instance attributes of the concept `physical person` of our legal entity ontology. The main areas of this interface are:

- The browsing area. To navigate through the whole ontology and to create new elements and modify or delete the existing ones.
- The clipboard. To easily copy and paste information between forms, so that similar ontology components can be created easily.
- The edition area. To insert, delete and update ontology terms (concepts, attributes, relations, etc.) with HTML forms, and tables with knowledge about existing terms. Figure 5 shows the attributes defined for the concept `physical person`: `age`, `birthday`, `deathday`, `first family name`, etc.

ODEDesigner eases the construction of concept taxonomies and ad hoc binary relations between concepts, and allows defining views to highlight or customize the visualization of fragments of the ontology for different users.

Concept taxonomies are created with the following set of predefined relations: *Subclass-Of*, *Disjoint-Decomposition*, *Exhaustive-Decomposition*, *Partition*, *Transitive-Part-Of* and *Intransitive-Part-Of*. Figures 3 and 4 show different views of our legal entity ontology in ODEDesigner.

[5] http://webode.dia.fi.upm.es/



Instance Attributes for Term *physical person.*

Instance Attribute Name	Description	Value Type	Cardinality	Measurement Unit	Precision	Value Interval
age		Integer	(1, 1)	year		0 -
birth day	The day when a person was born	Date	(1, 1)			
death day	The date when a person die	Date	(1, 1)			
first family name		String	(1, 1)			
first name		String	(1, 1)			
nationality		String	(0, N)			
second family name		String	(1, 1)			

Term Name	physical person
Instance Attribute Name	
Description	
Value Type	String

Fig. 5: Edition of an instance attribute with the WebODE ontology editor.

The WebODE Axiom Builder (WAB) is a graphical editor for creating formal axioms and rules, like the ones presented in table 7 and table 8. This editor aims at facilitating the creation of such components by domain experts who have not much experience with modelling in first order logic.

We now describe other ontology building services integrated in the ontology editor: the documentation service, ODEMerge, and the evaluation service. There are many other WebODE services (e.g. the OKBC-based Prolog inference engine, ODEClean, the ontology translation services, etc.) that will not be presented, since we think that they will not be specially useful for the readers for whom this paper is focused on.

The WebODE ontology documentation service generates WebODE ontologies in different formats that can be used to provide their documentation: HTML tables representing the METHONTOLOGY's intermediate representations described in section 2 and HTML concept taxonomies. In fact, the figures presented in such a section are part of WebODE screenshots.

The WebODE merge service (ODEMerge) performs a supervised merge of concepts, attributes, and ad hoc binary relations from two ontologies built for the same domain. It uses natural language resources to find the mappings between the components of both ontologies so as to generate the resulting merged ontology.

Finally, the WebODE workbench also provides the following ontology evaluation functions: the ontology consistency service and the RDF(S), DAML+OIL, and OWL evaluation services.

The ontology consistency service provides constraint checking capabilities for the WebODE ontologies and is used by the ontology editor during the ontology building process. It checks type constraints, numerical values constraints, and cardinality constraints, and verifies concept taxonomies (i.e., external instances of an exhaustive decomposition, loops, etc.).

The RDF(S), DAML+OIL, and OWL evaluation services evaluate ontologies according to the evaluation criteria identified by Gómez-Pérez [11]. They detect errors in ontologies implemented in these languages and provide suggestions about better design criteria for them.

4. Other methods and tools for ontology development

Basically, a series of methods and methodologies for developing ontologies have been reported in literature. In 1990, Lenat and Guha [16] published some general steps 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 [23] and the TOVE (TOronto Virtual Enterprise) project ontology [14] both in the domain of enterprise modeling, the first guidelines were proposed and later refined in [23].

At the 12th European Conference for Artificial Intelligence (ECAI'96), Bernaras and colleagues [3] presented a method to build an ontology in the domain of electrical networks as part of the Esprit KACTUS project. The METHONTOLOGY methodology appeared simultaneously and was extended in further papers [9, 10]. In 1997, a new method was proposed for building ontologies based on the SENSUS ontology [22]. Then some years later, the On-To-Knowledge methodology appeared within the project with the same name [20].

Concerning ontology tools' technology, it has improved enormously since the creation of the first environments. If we take into consideration the evolution of ontology development tools since they appeared in the mid-1990s, we can distinguish two groups[6]:

- Tools whose knowledge model maps directly to an ontology language. These tools were developed as ontology editors for a specific language. In this group we include: the Ontolingua Server [8], which supports ontology construction with Ontolingua and KIF; OntoSaurus [22] with Loom; and OilEd [2] with OIL first, later with DAML+OIL, and now with OWL
- Integrated tool suites whose main characteristic is that they have an extensible architecture, and whose knowledge model is usually independent of an ontology language. These tools provide a core set of ontology related services and are easily extended with other modules to provide more functions. In this group we can include Protégé-2000 [18], WebODE [1], OntoEdit [21], and KAON [17].

[6] In each group, we have followed a chronological order of appearance.

5. Conclusions

In this paper, we have shown how experts on the legal domain can develop their own ontologies following the ontology building methodology METHONTOLOGY and using the ontology engineering workbench WebODE. This methodology and tool have been successfully used by different groups for the development of ontologies in diverse domains. To illustrate how to use them, we have provided an example of how to develop an ontology about legal entities in Spain, adapting a taxonomy of legal entities elaborated by Breuker.

The main conclusion that we can transmit to the reader is that the broad experience on knowledge representation is not a necessary condition to build an ontology. Experts on the legal domain can have the initiative in the development of ontologies of their field with punctual help of experts on knowledge engineering. METHONTOLOGY allows modelling ontologies through graphical and tabular intermediate representations that can be understood by experts in one domain who are not deeply involved in the ontology field. Moreover, WebODE is a software platform that provides support to METHONTOLOGY, although it does not force to follow such methodology.

Finally, in section 4 we have presented other methods and tools so that readers can have the possibility of working according to other proposals.

Acknowledgments

This work has been supported by the project Esperonto (IST-2001-34373).

6. References

1. Arpírez JC, Corcho O, Fernández-López M, Gómez-Pérez A (2003) *WebODE in a nutshell*. AI Magazine, 24(3)-37-47
2. Bechhofer S, Horrocks I, Goble C, Stevens R (2001) *OilEd: a reasonable ontology editor for the Semantic Web*. In: Baader F, Brewka G, Eiter T (eds) Joint German/Austrian conference on Artificial Intelligence (KI'01). Vienna, Austria. (Lecture Notes in Artificial Intelligence LNAI 2174) Springer-Verlag, Berlin, Germany, pp 396–408
3. Bernaras A, Laresgoiti I, Corera J (1996) *Building and reusing ontologies for electrical network applications*. In: Wahlster W (ed) European Conference on Artificial Intelligence (ECAI'96). Budapest, Hungary. John Wiley and Sons, Chichester, United Kingdom, pp 298–302
4. Brickley D, Guha RV (2004) *RDF Vocabulary Description Language 1.0: RDF Schema*. W3C Recommendation. *http://www.w3.org/TR/PR-rdf-schema*
5. Chaudhri VK, Farquhar A, Fikes R, Karp PD, Rice JP (1998) *Open Knowledge Base Connectivity 2.0.3*. Technical Report. *http://www.ai.sri.com/~okbc/okbc-2-0-3.pdf*
6. Dean M, Schreiber G (2004) *OWL Web Ontology Language Reference.* W3C Recommendation. *http://www.w3.org/TR/owl-ref/*

7. de Hoog R (1998) *Methodologies for Building Knowledge Based Systems: Achievements and Prospects*. In: Liebowitz J (ed) Handbook of Expert Systems. CRC Press Chapter 1, Boca Raton, Florida
8. Farquhar A, Fikes R, Rice J (1997) *The Ontolingua Server: A Tool for Collaborative Ontology Construction*. International Journal of Human Computer Studies 46(6):707–727
9. Fernández-López M, Gómez-Pérez A, Juristo N (1997) *METHONTOLOGY: From Ontological Art Towards Ontological Engineering*. Spring Symposium on Ontological Engineering of AAAI. Stanford University, California, pp 33–40
10. Fernández-López M, Gómez-Pérez A, Pazos A, Pazos J (1999) *Building a Chemical Ontology Using Methontology and the Ontology Design Environment*. IEEE Intelligent Systems & their applications 4(1):37–46
11. Gómez-Pérez A (2001) *Evaluation of Ontologies*. International Journal of Intelligent Systems 16(3):391–409
12. Gómez-Pérez A, Fernández-López M, Corcho O (2003) *Ontological Engineering: with examples from the areas of knowledge management, e-commerce and the Semantic Web,* Springer-Verlag, New York.
13. Gómez-Pérez A, Juristo N, Montes C, Pazos J (1997) *Ingeniería del Conocimiento: Diseño y Construcción de Sistemas Expertos*. Ceura, Madrid, Spain
14. Grüninger M, Fox MS (1995) *Methodology for the design and evaluation of ontologies* In Skuce D (ed) IJCAI95 Workshop on Basic Ontological Issues in Knowledge Sharing, pp 6.1–6.10
15. IEEE (1996) *IEEE Standard for Developing Software Life Cycle Processes*. IEEE Computer Society. New York. IEEE Std 1074-1995
16. Lenat DB, Guha RV (1990) *Building Large Knowledge-based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley, Boston, Massachusetts
17. Maedche A, Motik B, Stojanovic L, Studer R, Volz R (2003) *Ontologies for Enterprise Knowledge Management*. IEEE Intelligent Systems 18(2):26–33
18. Noy NF, Fergerson RW, Musen MA (2000) *The knowledge model of Protege-2000: Combining interoperability and flexibility*. In: Dieng R, Corby O (eds) 12th International Conference in Knowledge Engineering and Knowledge Management (EKAW'00). Juan-Les-Pins, France. (Lecture Notes in Artificial Intelligence LNAI 1937) Springer-Verlag, Berlin, Germany, pp 17–32
19. Noy NF, Musen MA (2000) *PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment*. In: Rosenbloom P, Kautz HA, Porter B, Dechter R, Sutton R, Mittal V (eds) 17th National Conference on Artificial Intelligence (AAAI'00). Austin, Texas, pp 450–455
20. Staab S, Schnurr HP, Studer R, Sure Y (2001) *Knowledge Processes and Ontologies*. IEEE Intelligent Systems 16(1):26–34
21. Sure Y, Erdmann M, Angele J, Staab S, Studer R, Wenke D (2002) *OntoEdit: Collaborative Ontology Engineering for the Semantic Web*. In: Horrocks I, Hendler JA (eds) First International Semantic Web Conference (ISWC'02). Sardinia, Italy. (Lecture Notes in Computer Science LNCS 2342) Springer-Verlag, Berlin, Germany, pp 221–235
22. Swartout B, Ramesh P, Knight K, Russ T (1997) *Toward Distributed Use of Large-Scale Ontologies*. In: Farquhar A, Gruninger M, Gómez-Pérez A, Uschold M, van der Vet P (eds) AAAI'97 Spring Symposium on Ontological Engineering. Stanford University, California, pp 138–148
23. Uschold M, Grüninger M (1996) *Ontologies: Principles, Methods and Applications*. Knowledge Engineering Review 11(2):93–155
24. Uschold M, King M (1995) *Towards a Methodology for Building Ontologies*. In: Skuce D (eds) IJCAI'95 Workshop on Basic Ontological Issues in Knowledge Sharing. Montreal, Canada, pp 6.1–6.10