



OWL ontology use for terminology work

María Rosario Bautista-Zambrana

University of Málaga
Department of English, French and German Philology
Calle León Tolstoi 4, Facultad de Turismo, 29071 Málaga, Spain
mrbautista@uma.es

Keywords: *terminology, ontology, Web Ontology Language (OWL), ontology editor.*

Abstract

In this paper we revisit the question of the relationship between ontologies and terminologies, i.e. whether the former can be useful for building the latter. We review some of the approaches taken to address that topic and then construct a domain ontology for terminological purposes by means of the OWL-based ontology editor *TopBraid Composer Free Edition*. After showing how we have constructed the ontology, we analyze the results by focusing on five aspects: representation of conceptual relationships and characteristics; representation of linguistic relationships; use of abstract concepts; data categories that can be represented; and ontology display. Our results indicate that ontologies can be created for terminological purposes, but this comes with limitations.

1 Introduction

The concept of *ontology* comes from Greek philosophy and is usually defined as the “subject of study in philosophy that is concerned with the nature of existence” (Summers, 1995: 989). As Sowa (1999) states, “its primary task, as it was practiced by its founder Aristotle, is to bridge the gap between what exists and the languages, both natural and artificial, for talking and reasoning about what exists”. In a simplified way, we can assert that it studies the entities that exist and how they are related to one another, in order to reason about them.

In the past decades the concept of *ontology* was adopted by computer science, in order to develop “formalized, semantic, and logic-based models, which can easily be implemented in computer systems” (Øhrstrøm *et al.*, 2005: 425). The study of ontologies was specially undertaken by artificial intelligence, which led to the birth of the subfield of ontological engineering. In this field *ontology* is usually defined as “an explicit specification of a conceptualization” (Gruber, 1993: 199). This author understands *conceptualization* as “an

abstract, simplified view of the world that we wish to represent for some purpose” (*idem*). Consequently, an ontology describes a certain abstract vision of the world in an explicit, formal way. This vision is made up of “objects, concepts, and other entities that are presumed to exist in some area of interest and the relationships that hold them” (Genesereth & Nilsson, 1987 *apud* Gruber, 1993: 199). Nowadays ontologies play a crucial role in the Semantic Web, a movement led by the World Wide Web Consortium (W3C) which encourages the inclusion of semantic content in web pages.¹

At this point it is interesting to refer to the components of an ontology, especially from the point of view of ontological engineering. According to Corcho et al. (2005: 144-145), any ontology is made up of concepts, relations, instances, constants, attributes, axioms and rules. These authors describe these elements as follows:²

- Concepts, also called classes, represent ideas about the physical or abstract objects that constitute a domain. As the authors point out, concepts “are usually organized in taxonomies through which inheritance mechanisms can be applied” (Corcho et al., 2005: 144). In this way, a class can be divided into subclasses that represent more specific concepts: for instance, an ontology can contain the class `means of transport`, and this can further contain the subclass `car`.
- Relations, or relationships, represent a type of association between concepts of the domain. The majority of relations link two concepts, so they are called binary relations.
- Instances represent individuals or specific elements of an ontology. So, for example, the concept `car` can be instantiated as “2001 Ford Focus”.
- Constants are numerical values that do not change during much time. For example, the minimum number of nights in a hotel is one.
- Attributes, also called properties, describe properties of concepts and of instances. The authors distinguish two types of attributes: “class attributes” and “instance attributes” (*idem*: 145). The former describe concepts and take their values in the concept where they are defined. The latter describe instances and take their values in them.
- Formal axioms are logical expressions that are always true and are normally used to specify constraints in the ontology.
- Rules are generally used to infer knowledge in the ontology, such as attribute values.

Two more disciplines that have benefitted from the formal specification provided by ontologies are natural language processing (NLP) and terminology. Beale *et al.* (1995: 4) single out the uses of ontologies for NLP:

An ontology for NLP purposes is a body of knowledge about the world (or a domain) that a) is a repository of primitive symbols used in meaning representation; b) organizes these symbols in a tangled subsumption hierarchy; and c) further interconnects these symbols using a rich system of semantic relations defined among the concepts. [...] The ontology must be put into well-defined relations with other knowledge sources in the system. In this application, the ontology supplies world knowledge to lexical, syntactic and semantic processes, and other microtheories.

We would like to emphasize the statement “the ontology supplies world knowledge to lexical, syntactic and semantic processes,” as it indicates that the concepts in the ontology can be connected to lexical units, and thus a knowledge base made up of concepts organized in

¹ http://en.wikipedia.org/wiki/Semantic_Web

² We have provided our own examples.

hierarchies could be enriched with linguistic information. This idea was borrowed as well by terminology.

Terminology is mentioned in this paper with a twofold meaning (Valeontis and Mantzari, 2006: 1):

1. the scientific field pertaining to the study of relations between concepts and their designations (terms, names and symbols) and the formulation of principles and methods governing these relations in any given subject field; and the task of collecting, processing, managing and presenting terminological data in one or more languages, as well as
2. the set of terms belonging to the special language of a specific subject field.

As these authors explain (*idem*):

Fundamental for the theory of terminology is the distinction between objects, i.e. entities in the external world, concepts, which are the units of knowledge that constitute the mental representations of objects, and designations of concepts, which can be terms, names and symbols. Concepts are further determined by means of the relations they have to other concepts, as well as by definitions, which constitute the descriptive, metalinguistic denotation of concepts.

These components (objects, concepts, designations of concepts, relations, definitions) are similar to those of ontologies, and that is why many researchers have explored the connection between ontologies and terminologies, and how they can be beneficial to one another.³ We will specifically focus in this paper on using ontologies for terminology. A number of authors have dealt with this topic: for instance, Feliu et al. (2002), Temmerman and Kerremans (2003), Cabré (2004), Moreno Ortiz (2008), Faber et al. (2009), Leonardi (2012), or Durán-Muñoz and Bautista-Zambrana (2013).⁴

The connection between ontologies and terminologies is explained by Moreno Ortiz (2008: 3) as follows: the terminologist works with concepts; he draws up and defines in a systematic way the conceptual relationships that exist among those concepts, creates taxonomies and specifies which lexical units are used, in the diverse languages involved, to make reference to those concepts. Therefore, the author adds, it seems that a formalized, explicit and standardized conceptual representation system should facilitate the terminologist's work, at least partly.

However, as other researchers claim, the relationship between ontologies and terminologies is not so apparent. For instance, according to Hirst (2009: 8), "a lexicon, especially one that is not specific to a technical domain [...], is not a very good ontology."⁵ L'Homme and Bernier-

³ In this respect, Grabar et al. (2012: 376) point out that "because of the proximity in their semantic content and application contexts, terminologies and ontologies may be better thought of as part of a continuum rather than completely distinct types of artifacts."

⁴ It is worth noting that ontologies have become an important object of study for a number of terminology-oriented research groups and that several periodical conferences have been dealing with this area of expertise for years now: the Terminology & Ontology: Theories and Applications Conference (TOTh), the International Conference on Terminology and Artificial Intelligence (TIA) and the Terminology and Knowledge Engineering Conference (TKE). (cf. Durán-Muñoz and Bautista-Zambrana, 2013).

⁵ However, as Hirst elaborates on this issue, he goes on to state that "it is possible that a lexicon with a semantic hierarchy might serve as the basis for a useful ontology, and an ontology may serve as a grounding for a lexicon. This may be so in particular in technical domains, in which vocabulary and ontology are more closely tied than in more general domains" (Hirst, 2009: 14).

Colborne (2012) agree with this idea and state that ontologies and terminologies are dealing with different objects: “ontologies are dealing with terms = class labels or with terms = linguistic expressions listed in annotations; dictionaries are dealing with terms = form + meaning” (*idem*: 396). In this way, the authors point out that “trying to exchange contents from one resource to the other will inevitably result in making several adjustments (in the ontology or in the dictionary), unless one compromises on the very nature of the objects that are represented in them” (*idem*: 398). For her part, Gromann (2013: 427) states that “terminologies and ontologies cannot be treated equivalently as they differ from a syntactic, semantic, and pragmatic perspective”; as she explains, the most relevant syntactic difference is the refined modeling of semantic relations and fine-grained natural language contents of terminologies, which have no corresponding elements in the ontology. From a semantic point of view, “ontologies realize the extensional definitions, while terminologies fully focus on intensional definitions of concepts” (*idem*: 426). On a pragmatic level, “terminologies are interested in how terms are used in specialized communication, while ontologies seek to draw inferences on the specific state of a certain domain of discourse” (*idem*: 426).

Taking the previous ideas as a background (commonalities and differences between ontologies and terminologies), we will describe the work that we have carried out in this field, namely constructing an ontology for terminological purposes, and we will show which aspects we have been able to portray in our ontology, and which ones not, so as to determine whether ontologies can be (properly) used for terminology work. Specifically, we set out to check which possibilities or obstacles we have encountered when dealing with the following aspects:

- Representation of conceptual relationships and characteristics.
- Representation of linguistic relationships (synonymy, near synonymy, collocations).
- Use of abstract or *artificial* concepts to better structure the domain.
- Data categories that can be represented.
- Ontology display.

The paper is organized as follows. Section 2 deals with how we have built our ontology; we explain the methodology that we have used and describe the implementation by means of the application *TopBraid Composer*. Section 3 discusses the results and offers some concluding remarks.

2 Creation of the ontology

Creating an ontology requires that we follow some ordered and defined steps, that is, to adopt a certain methodology. There are many methodologies for constructing ontologies, some of them devised for ontological engineering and some thought out for terminology. However, we consider that all of them have activities in common. In this case, we have opted for an ontological engineering methodology.

As a general rule, every methodology consists of a series of development-oriented activities: the most common are specification, conceptualization, formalization and implementation (Gómez-Pérez *et al.*, 2004). These authors describe them as follows: Specification is about determining why the ontology is being built, and what its intended uses and final users will be. Conceptualization consists in structuring the knowledge about a domain by means of significant models, on the knowledge level. It implies creating a so-called *conceptual model*, which is a representation of the expert’s knowledge, external to the computer, by means of non-

computable structures that describe the problem and its solution (Gómez Pérez *et al.*, 1997; Fernández-López *et al.*, 1997). A conceptual model is usually made of intermediate representations, based on graphics and tables that can be understood by both domain experts and ontology developers. On the other hand, formalization transforms the conceptual model into a formal or semi-computable model. In this way, as Gómez Pérez *et al.* (1997) explain, the problem is modeled from the point of view of the system; they add that this is carried out through formalisms or knowledge representation techniques, some of which are based on concepts (such as frames), some on relationships (such as logic and semantic networks), and some on actions (such as scripts). Finally, implementation involves building computable models, that is, computer-readable ones, by means of an ontology language; according to Clark *et al.* (2004), “(a)n ontology language is a means by which one can formally describe a knowledge domain, with the goal of enabling computers to provide various kinds of reasoning services about that domain, and about the knowledge described by an ontology for that domain.”

We must point out, however, that there are computer applications, such as *Protégé*, that automatically translate conceptual models into formal models described by means of ontological languages, so sometimes it is not necessary to carry out the formalization activity *per se*.

As regards the terminology field, against the background of growing interest in ontologies as a base for terminology work, some research groups have devised and created editors to build ontology-oriented terminological resources: some examples are *OntoTerm*, *Multilingual Categorisation Framework Editor* or *TERMINAE*. However, as Durán-Muñoz and Bautista-Zambrana (2013) point out, there are not freely available tools for building ontology-based terminological resources that enable the inclusion of multilingual designations or that support enough features so as to make a difference from traditional terminology editors (by offering, for instance, clear organization of specialized knowledge, systematic and coherent definitions, representation of multidimensionality, dynamicity, addition of multilingual information, among other features). Moreover, the existing ontology-based terminological editors might not be well suited to different methodologies and conceptual models (cf. Bautista Zambrana, 2013: 327).

The alternative is not straightforward either: it still remains difficult to find standard ontology editors that meet all the features needed to properly construct an ontology aimed at terminologists, or translators. Although, as we have observed, there are many computer applications for building ontologies, these are mainly intended for ontological engineering. Still, we propose to explore their possibilities and check whether they are suitable for terminology work.

This study was carried out taking into account the needs of a terminological project on package travel⁶ (cf. Bautista Zambrana, 2013); it involved the phases of specification and conceptualization, as well as support activities such as knowledge acquisition.

2.1 Specification activity

The specification activity consisted in determining the motivation we had to build the ontology, which uses and end-users it would have. In this regard, it is an ontology aimed at translators, terminologists and other professionals that need multilingual documentation, and is thought out

⁶ As regulated by the *Council Directive 90/314/EEC of 13 June 1990 on package travel, package holidays and package tours*.

as a base for the creation of a multilingual ontoterminological data bank or dictionary on package travel, for the Spanish, English and German languages. This implied that the ontology would have an independent, common backbone, composed of concepts, properties, and relationships among concepts, valid for all the languages involved, and that every concept, property and relationship would be lexicalized by means of their corresponding terms in Spanish, English and German.

2.2 Conceptualization activity

The conceptualization phase required some previous support activities, especially those aimed at acquiring conceptual and linguistic knowledge; to this end, we compiled a multilingual text corpus (Spanish-English-German) on package travel, containing laws and contracts (terms and conditions), in order to extract the most important terms (and therefore their underlying concepts) of this domain, and the conceptual relationships that they hold with each other. Translation equivalents for the three languages were also detected. In this sense, it must be pointed out that our approach to extracting concepts was mainly semasiological, i.e. we adopted a bottom-up method to extract the terms that would later populate the ontology; the relationships were extracted using bottom-up and top-down methods.

Once we had collected all this information together, we were able to proceed to the conceptualization phase.⁷ It consisted in organizing and structuring the knowledge acquired in the abovementioned activity, by means of external representations, independent from the knowledge representation paradigms and ontology languages that we would later use to implement the ontology; in other words, we represented the domain under study —package travel— through concepts that were comprehensible by both domain experts and ontology developers, that is, by means of diagrams and tables; the result was the conceptual model of the domain. For instance, one of the steps required to construct taxonomies from the concepts extracted from the corpus. Figure 1 illustrates a small taxonomy for the English language.

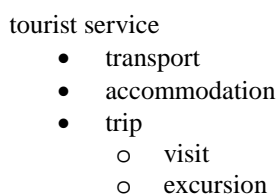


Figure 1 *Taxonomy as part of the conceptualization phase.*

2.3 Implementation

The next step, formalization, was replaced by the subsequent activity, implementation, since we decided to employ a computer tool to build the ontology. We needed a tool that was user-friendly, well-documented, and that had the capability of exporting the produced ontology into

⁷ Our work was partly based on the METHONTOLOGY methodology (Gómez-Pérez et al., 2004).

other formats. Thus, in order to carry out this task, we pre-selected three ontology editors based on their general relevance, usefulness, usability and adaptability: *Protégé*, *NeOn Toolkit* and *TopBraid Composer*.

In this section, with the aim of finding an appropriate ontology editor, we compare the three selected tools —*Protégé*, *NeOn Toolkit* and *TopBraid Composer*— by focusing on the aspects that are most relevant for our purposes: simplicity (clarity, user-friendliness), documentation, user community and export capabilities.⁸

Protégé

*Protégé*⁹ is one of the most used and best-known tools for creating ontologies. It is a free and open-source application, with three main versions: *Protégé-OWL*,¹⁰ *Protégé-Frames*¹¹ and *WebProtégé*. We decided to try the OWL version (versus the Frames one), given the rise of this language as an interchange format for semantic data, and owing to the greater amount of documentation on the Web, as well as of browsers capable of reading ontologies written in OWL. *Protégé-OWL*, in turn, is divided into two large groups of versions, those which start from 3.X and those which do it from 4.X. We decided to try the first, in particular the version 3.4.6, since it is more stable, is based on the more widespread (up to now) OWL 1.0, and supports OWL and RDF(S)¹².

By means of *Protégé-OWL* we can represent concepts, relationships and properties, and moreover each entity can be documented with labels in all the desired languages (see Figure 2). Descriptions for each entity can be added as well, in any language. The ontology can be created in a great variety of formats (OWL/RDF, RDF, Experimental XML), what facilitates its subsequent use in other applications, and in general its sharing and interoperability with other platforms.

The large amount of manuals and tutorials about *Protégé* on the Web are a great advantage to take into account.

⁸ These are mainly extrinsic aspects. The three applications are internally quite similar, since they are based on OWL.

⁹ <http://protege.stanford.edu/>

¹⁰ OWL, or Web Ontology Language, is based on the DAML-OIL web ontology language, which in turn builds on the RDF and RDF(S) languages. More information at <http://www.w3.org/TR/owl-features/>

¹¹ This version is based on frames, a knowledge representation paradigm. The official website of *Protégé* describes it as follows: “In this model, an ontology consists of a set of classes organized in a subsumption hierarchy to represent a domain's salient concepts, a set of slots associated to classes to describe their properties and relationships, and a set of instances of those classes - individual exemplars of the concepts that hold specific values for their properties” (see <http://protege.stanford.edu/overview> for more information).

¹² RDF(S) or RDF Schema is a semantic extension of the ontology language RDF. We can find many of its components in OWL. More information at <http://www.w3.org/TR/rdf-schema>

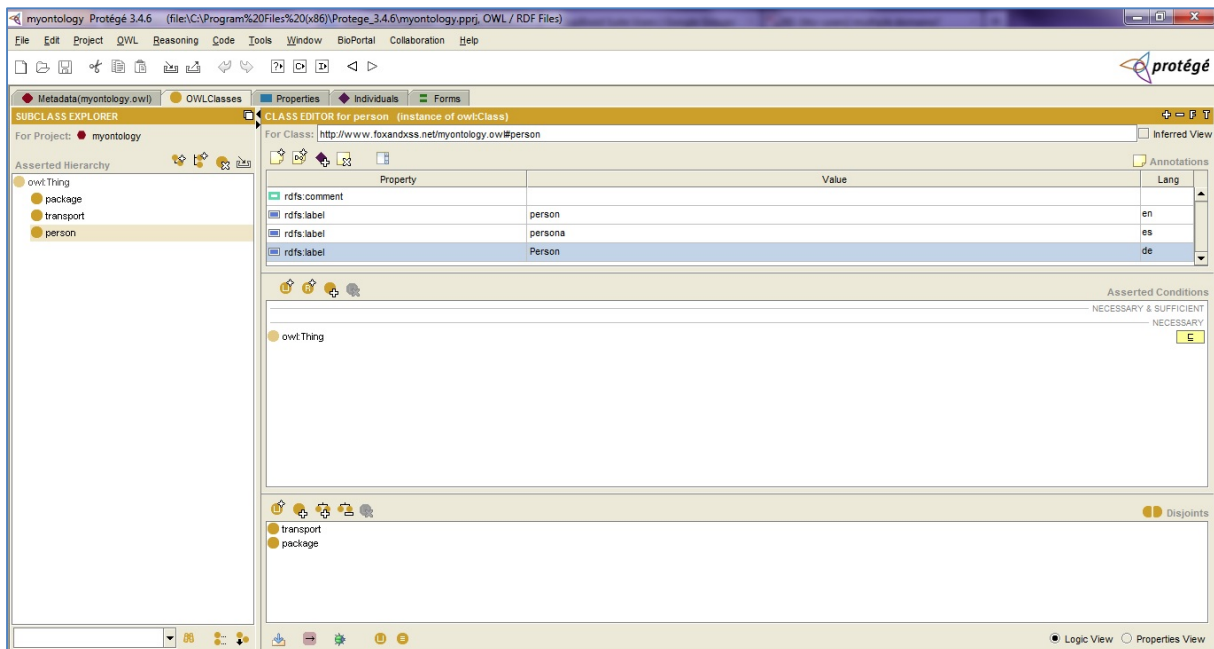


Figure 2 Class editor of Protégé 3. 4. 6.

Another remarkable aspect of *Protégé-OWL* is its large community, with more than 240,000 registered users: there are various official mailing lists (which cover the different versions of the program), as well as an extensive wiki with documentation for users and developers, and relevant links about the program.¹³

NeOn Toolkit

*NeOn Toolkit*¹⁴ is a much more recent and less-known tool than *Protégé*. It was created within the NeOn Project (European Commission's Sixth Framework Programme) and has had a number of versions, the last being 2.5.2, as of December 2011. This one was the version used for our comparison. *NeOn Toolkit* is aimed at creating ontologies with OWL, and for that purpose it includes a complete editor, which supports OWL 2. The editor is quite intuitive and allows creating concepts, properties and relationships in a relatively easy way; likewise, labels can be added to each one of the cited entities. We can state that, broadly speaking, it is easier to manage than *Protégé*, owing partly to the lower number of options (see Figure 3). As in the case of *Protégé*, ontologies can be saved in several formats (in this case, OWL, RDF and RDF(S)).

¹³ http://protegewiki.stanford.edu/wiki/Main_Page>

¹⁴ http://neon-toolkit.org/wiki/Main_Page



However, the program has less documentation than other analyzed applications and does not have a considerable user community (for example, in the form of a forum) where one can ask questions or look for information; there exists, nevertheless, a mailing list for these purposes.

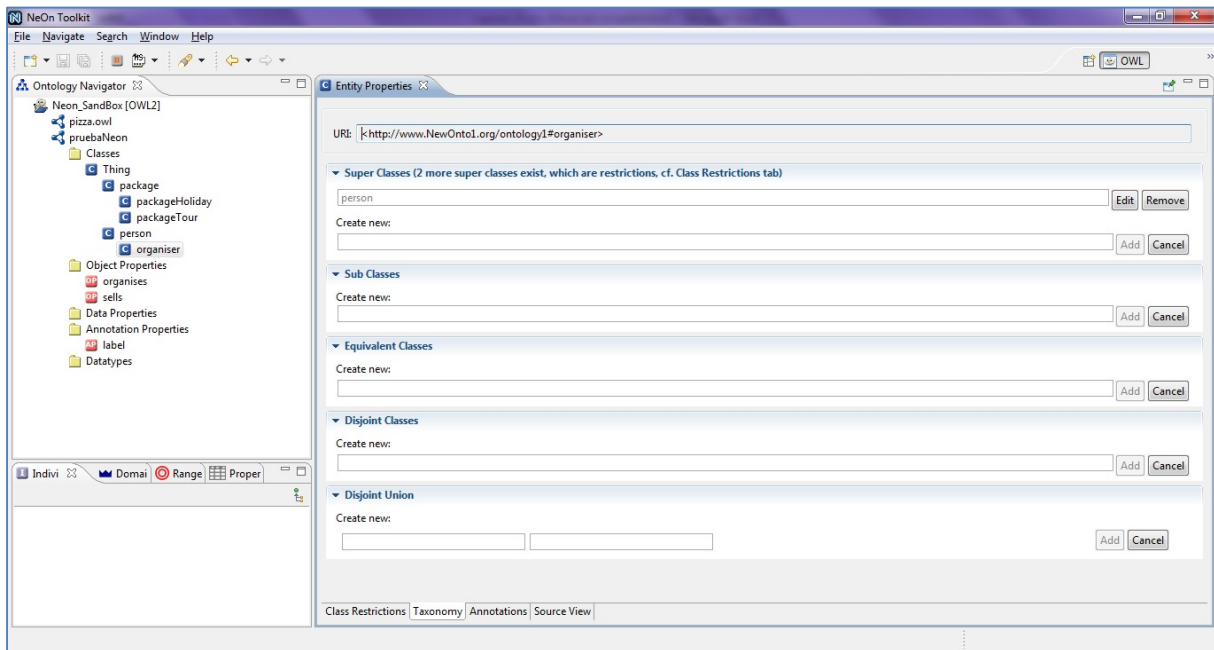


Figure 3 Taxonomy editor in NeOn Toolkit 2. 5. 2.

TopBraid Composer

Finally, we will deal with *TopBraid Composer Free Edition* (hereafter *TopBraid*). This program of the company TopQuadrant is remarkable for its usability and the help that its extensive documentation provides: basic guides (TopQuadrant, 2011, *Getting started Guide*), manuals, and a Google group page containing numerous frequent questions, together with answers from the users themselves and the program developers.¹⁵

One of its key features is, as mentioned above, the ease of use: the application is quite intuitive, while we can find detailed explanations of every procedure (such as creation of classes, properties and relationships) in the basic guides and manuals. Like *Protégé*, *TopBraid* is oriented towards logic work (inferences, reasoning tasks), but one can also use the program and benefit from its many features without making use of those functionalities.

¹⁵ <<https://groups.google.com/forum/?fromgroups#!forum/topbraid-users>>

As in the other applications, labels can be attached to each concept, relationship or property (see class editor in Figure 4). In this respect, a disadvantage is that the corresponding language code has to be added manually behind the corresponding designation or definition, by writing, for instance, “{ @es}” (the quotation marks are ours).

We must also emphasize the capacity of the application to export into other formats, although it offers fewer alternatives than other programs: Turtle, XML/RDF abbreviated, XML/RDF, N-Triple and Java.

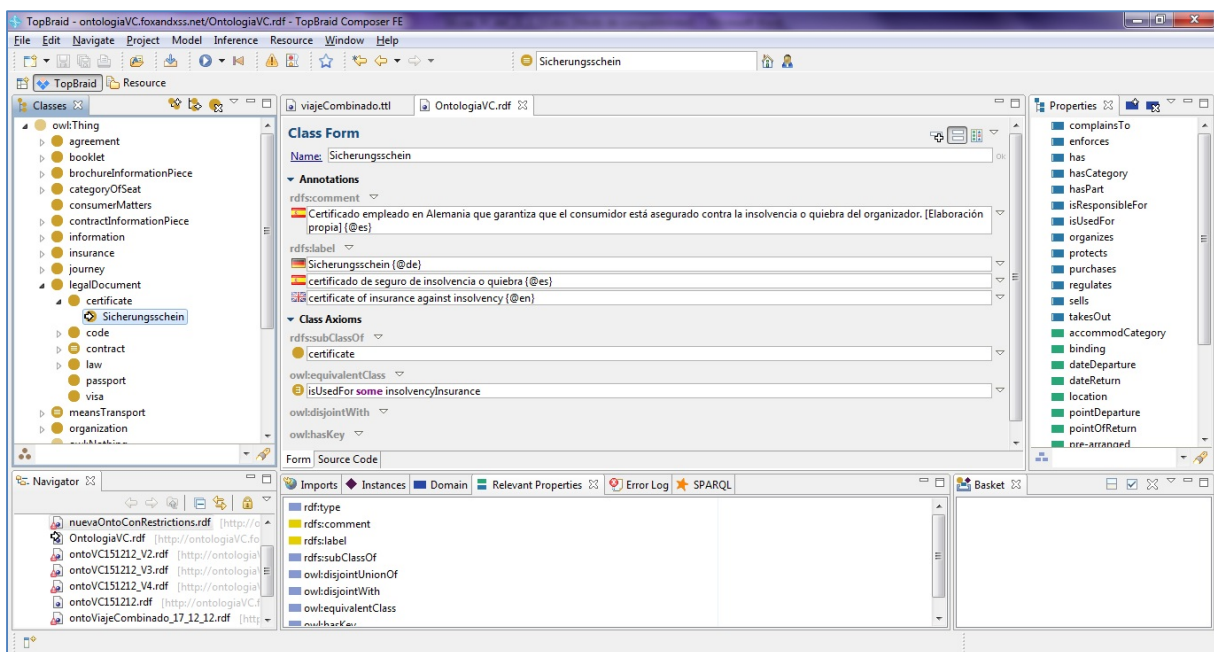


Figure 4 Class editor of TopBraid Composer.

Next, we offer a summary table of the main advantages (or disadvantages) brought by each program. We have assessed the main aspects cited so far (usability or simplicity, that is, clarity and intuitiveness; guides or documentation; community; and export capabilities), by giving a score from 0 (lowest) to 5 (highest), according to the user experience offered by each program.¹⁶

¹⁶ In the table, “a” stands for usability; “b” for documentation; “c” for community; and “d” for export capabilities.

Aspects →	a	B	c	d	Total score
Programs ↓					
<i>Protégé</i>	3	4	5	5	17
<i>NeOn Toolkit</i>	4	3	1	5	13
<i>TopBraid Composer</i>	4	5	5	4	18

Table 1 Comparison and scores for characteristics of ontology editors.

As we see, *TopBraid* obtains the highest score, followed very closely by *Protégé*. However, we have decided to select *TopBraid*, because we have found its ease of use and clarity very advantageous; in addition, it complies with the criteria of exporting to formats that allow working later on with other programs. The only drawback we have detected is the fact that it is a commercial application, since we would have preferred to employ an open-source program.

In the next section we will tackle the implementation of the ontology by means of *TopBraid*.

2.3.1 Ontology implementation by means of *TopBraid Composer*

The last activity in the process of constructing an ontology is implementation, which consists, as we explained above, in building computable models (readable by computer) in an ontology language. In other words, it involves formally representing previously collected concepts. This activity, as pointed out, can be carried out manually, i.e., by writing the ontology with the corresponding code of the chosen language, or by means of a computer application. We have opted for the second option, and as we concluded in the previous section, it was determined that the most appropriate program was *TopBraid Composer Free Edition*. In this way, this section will focus on explaining how we have transferred the conceptualization obtained in the previous activity to a computable model, by means of the aforementioned application.

Our aim is to construct a domain ontology, aimed at translators and terminologists who need conceptual and linguistic documentation on package travel. In this way, the ontology will be made up of concepts, relationships among those concepts and some relevant properties. The relationships will be hierarchical and associative. Each entity is to be accompanied by linguistic labels in English, Spanish and German. The labels used to lexicalize the concepts will thus constitute the terms of our specialized domain. A definition in Spanish, with an indication of its source, is to be added to each concept.

The first step is to create a project in *TopBraid*. For that purpose we have given it a name, “OntologiaVC”. Inside the project, we have created a RDF file in RDF/OWL format, “OntologiaVC.rdf”. This file will be our ontology.

Once the project and the file have been created, we can start working on the ontology. As we can see, there are different working sections (Figure 5): on the left side, we can find the concept tree; in the middle, we can see the class editor (or the relationship or property editor, as

appropriate); on the right side, the list of properties¹⁷ (which contains the relationships and intrinsic attributes).

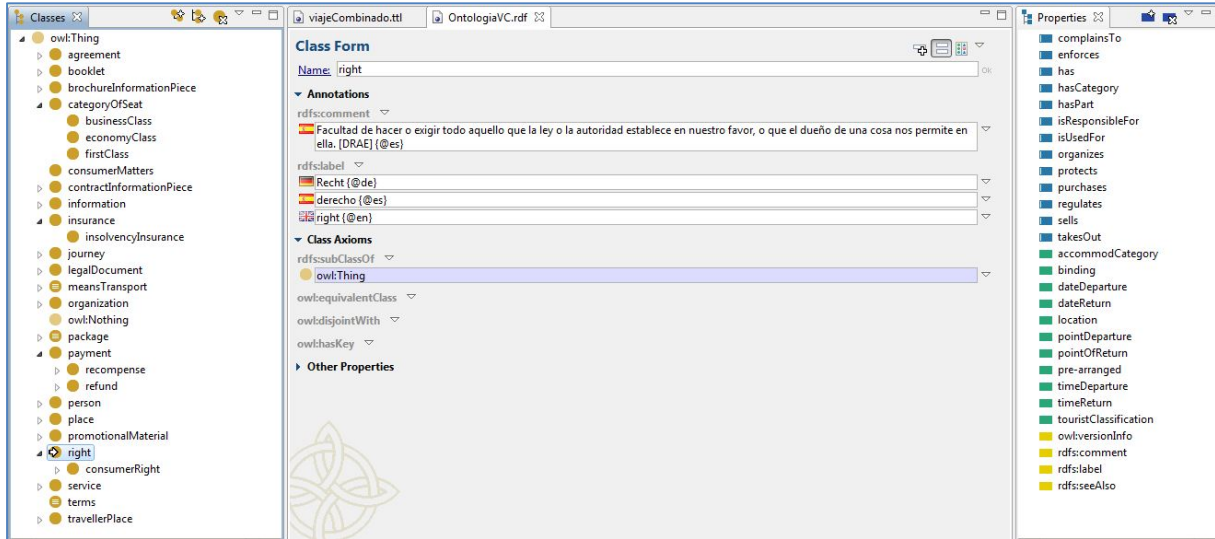


Figure 5 Working sections in TopBraid.

The first step consists in representing all the classes selected in the previous phase of conceptualization. To that effect, *TopBraid* contains a class tree and a class editor, which we show next (Figure 6):

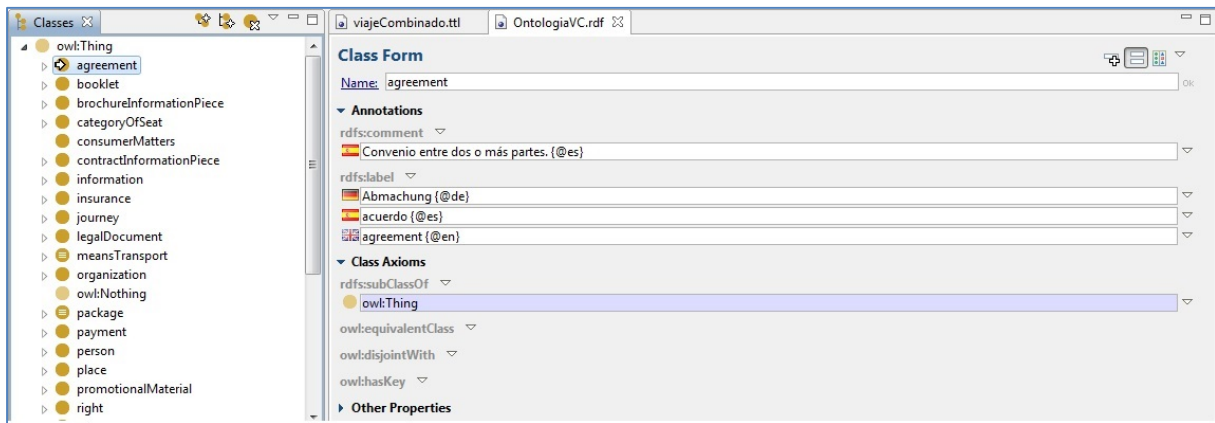


Figure 6 Class tree and class editor in TopBraid.

¹⁷ In OWL, both relationships and intrinsic attributes are called *properties*.

In the class editor there is a form where we can include the name of the class: it is specifically the concept identifier, not a linguistic designation for the concept. However, although we could have opted for naming the concepts as a number or any character string, we decided to use the corresponding terms in English, in many cases as an abbreviated form, without spaces: for instance, *legalDocument*.

Regarding the way of implementing the lexicalizations associated to each concept, there exist several methods. Montiel (2011: 203-210) sets out the three main ones:¹⁸

- Including multilingual labels in the ontology: it is the most widespread modeling option and consists in making use of the labeling functionality of RDF(S) and OWL.
- Combining the ontology with a mapping model: It consists in mapping, establishing correspondences or equivalences among conceptualizations in different languages.
- Associating the ontology with an external linguistic model: The ontology entities are linked to linguistic data stored outside of the ontology. It is the case of the model put forward by Montiel (2011: 232), *Linguistic Information Repository (LIR)*, which consists of a linguistic layer in different natural languages that captures the conceptual knowledge represented in a specific domain ontology, so it provides the conceptual model with linguistic information. Another remarkable example of external linguistic model, intended to model lexica and machine-readable dictionaries, is Lemon.¹⁹

In our case, we have opted for using the labeling functionality provided by RDF(S) (and included in OWL), since it allows assigning linguistic labels to concepts, relationships and properties; in addition, we consider that this functionality can adapt well to the domain conceptualization we have performed and to the divergences that may arise both on the conceptual level (different categorizations of reality in different cultures) and on the linguistic level (synonymy and polysemy). In particular, these designations or lexicalizations can be added in *TopBraid* in the Annotations section, under `rdfs:label`. By means of this label, RDF(S) allows us to add designations, in all the desired languages, to each concept. We can see an example in the following Figure:

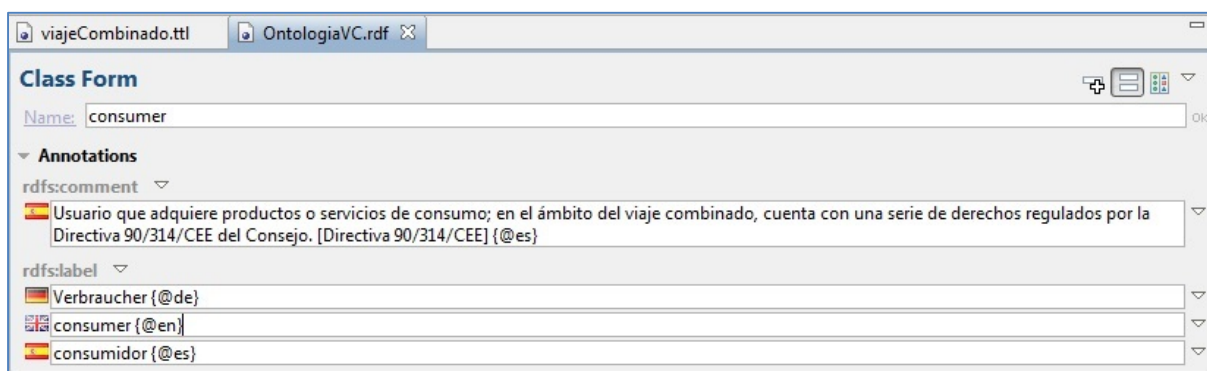


Figure 7 Labels in the class editor of TopBraid.

¹⁸ Although these methods are generally applied to ontology localization, we consider they can also be used to include terminological information in ontologies.

¹⁹ <http://lemon-model.net/index.php>

Next, the relationships were represented. These can be implemented in OWL by means of the so-called *object properties*, which represent connections between two classes of the same ontology. With that aim, in the Properties section, on the right side of the interface, we created each relationship or object property for our ontology. Like classes, relationships are identified by a unique name given by the user, to which different designations can be associated, in the desired languages. For example, we added the meronymic relationship *hasPart* and associated it with the following linguistic equivalents: *tiene parte* (Spanish), *has part* (English) and *hat Teil* (German). We can observe it in the following screenshot:

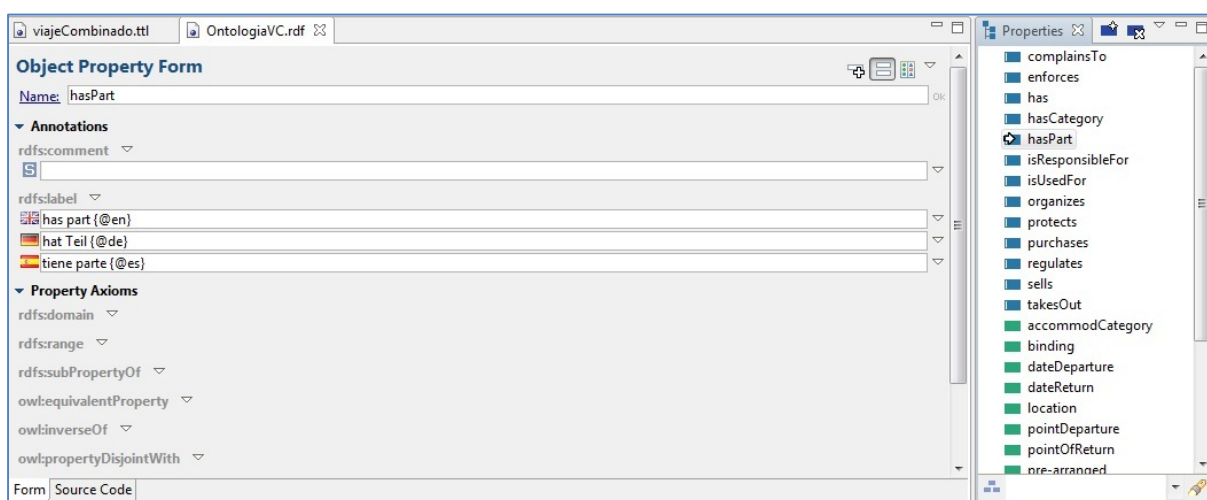


Figure 8 Relationship editor in TopBraid.

The implementation of the object properties in OWL posed a problem for the correct representation of the relationships we wanted to reflect in the ontology. The reason is that the primary way of establishing which is the first part of the relationship and which the second consists in specifying the *domain* (first part, expressed in OWL by the label *rdfs:domain*) and *range* (second part, expressed by *rdfs:range*). For instance, in the relationship contract *hasPart* terms, contract is the domain and terms is the range.²⁰

However, given that object properties are independent from classes, if we use again *hasPart* with a different domain and range (for example, *cabin hasPart berth*), it will be inferred that any instance of *cabin* is also one of *contract*, and vice versa; the same would be the case for *terms* and *berth*. Obviously, we do not want to have those inferences in our ontology; on that account we had to search for other methods of representing the desired relationships. There are two possible solutions: in the first place, we can use operators such as OR when specifying the respective domain or range. We can draw an example from the *TopBraid manual* (TopQuadrant, 2011: 26):

²⁰ We will use another font to distinguish the concepts (as well as the relationships and properties) from its respective linguistic designations.

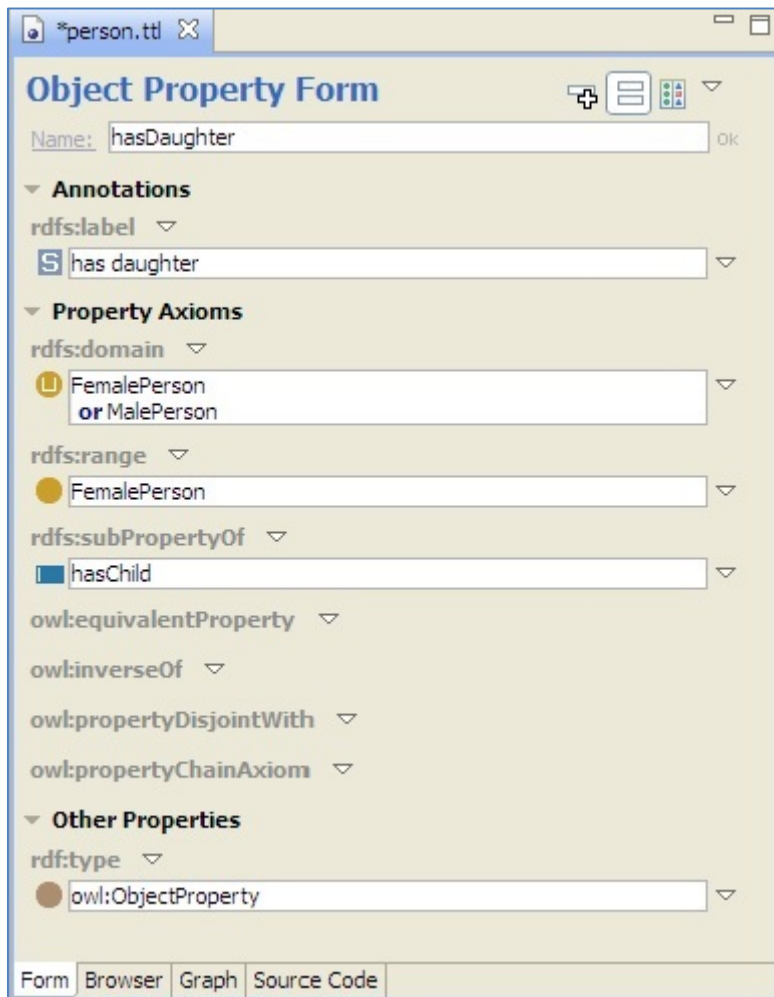


Figure 9 Use of the OR operator.

As we can see, the domain field of the form contains the expression `FemalePerson or MalePerson`, which constitutes the relationship `FemalePerson or MalePerson hasDaughter FemalePerson`. This means that either a woman or a man can have a female daughter. This solution is valid when we deal with very specific relationships, such as `hasDaughter`, since it is acceptable for the domain (the first part of the relationship) to be either a woman or a man. However, it is not valid when we wish to apply it to diverse concepts not related to one another, as in the case of `hasPart`. This relationship is applicable to multiple classes of different nature, so it is necessary to resort to the second solution: it consists in employing the so-called *OWL restrictions*. They bring the advantage of defining only classes; in contrast, object properties only define relationships, in such a way that their values (domain, range) remain always true, irrespective of where the relationship is being used. For their part, OWL restrictions can be used to restrict the individuals or instances that belong to a class, by

declaring, with the statements `rdfs:subClassOf` and `owl:equivalentClass`, the following restrictions (TopQuadrant, 2011: 41):

- Quantifier restrictions: *allValuesFrom* and *someValuesFrom*.
- Cardinality restrictions: *minCardinality*, *cardinality* and *maxCardinality*.
- *hasValue* restrictions.

An example of how restrictions work is the following (TopQuadrant, 2011: 41): taking the class `USCitizen` as an example, if we declare under `rdfs:subClassOf` that the nationality property *hasValue* ‘USA’, this means that `USCitizen` is a subclass of all things for which the value of the nationality property equals (*hasValue*) ‘USA’.

On the contrary, if we declare the same restriction in the field `owl:equivalentClass` (that is, by writing nationality *hasValue* ‘USA’ in that space), this means that the class `USCitizen` is equivalent to all things for which the value of nationality property equals (*hasValue*) ‘USA’. From this we can draw two inferences:

- if it is known that an individual is a US Citizen, it can be inferred that his nationality is ‘USA’, and
- if it is known that an individual’s nationality is ‘USA’, it can be inferred that he is a US Citizen.

In our case, we have opted for utilizing `owl:equivalentClass` declarations, with the quantifier restriction *someValuesFrom* (represented by *some* in the OWL code), which denotes that a class has at least one relationship to an instance of another class. We can see an example of this in the form corresponding to the class `organizer` (Figure 10).

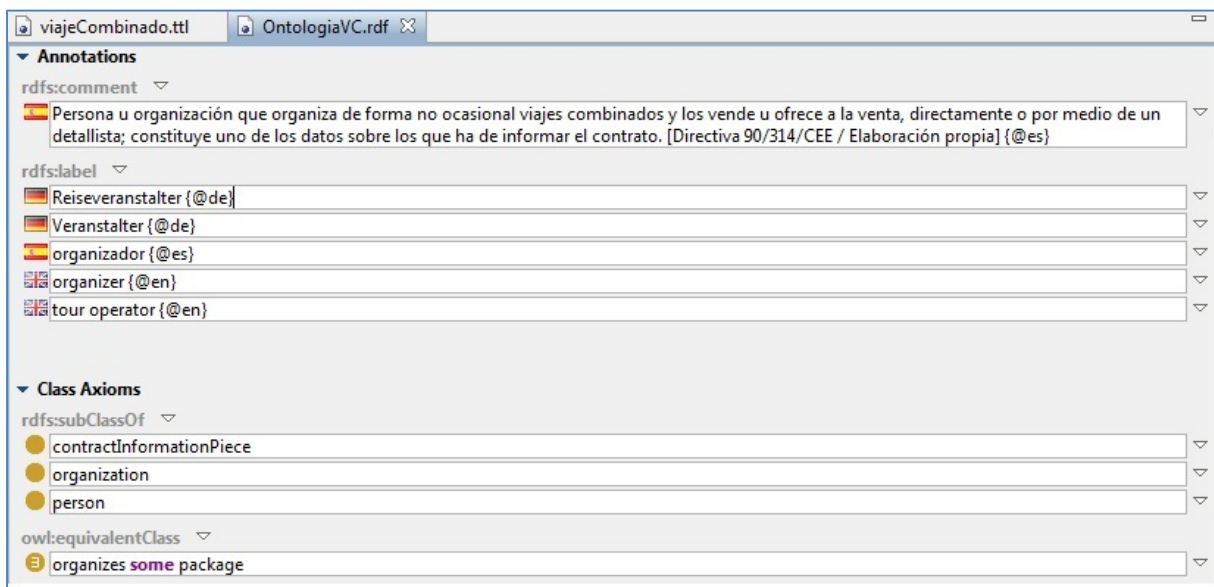
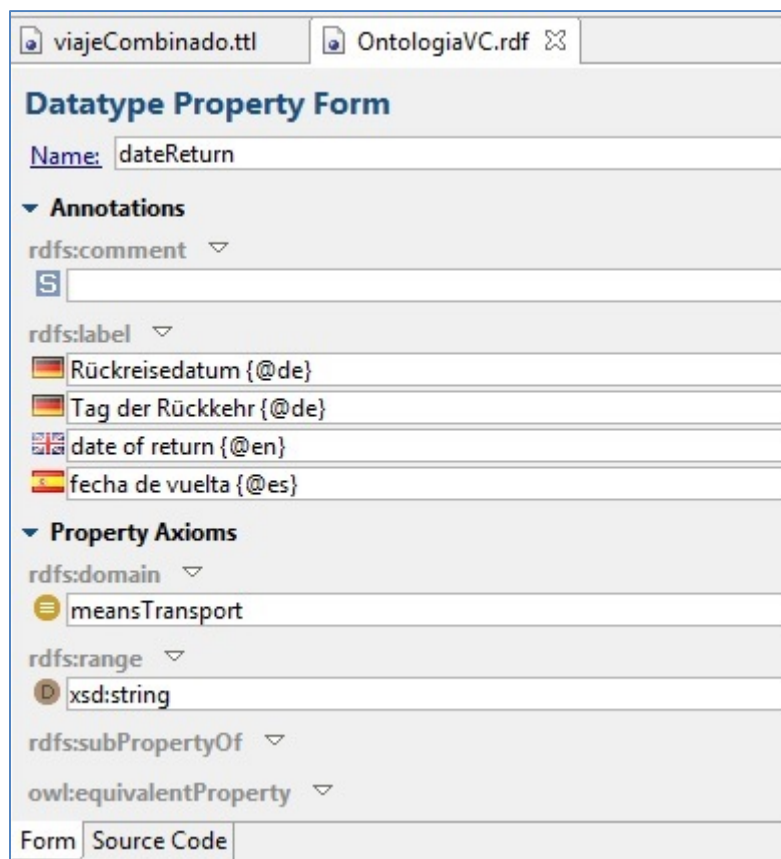


Figure 10 *Formulating relationships for the class organizer.*

By means of the declaration we have made under `owl:equivalentClass`, we are asserting that `organizer` equals all the things that organize at least one travel package. Therefore, we can infer that if an individual organizes any travel package, it must be an organizer.

Finally, the properties were implemented. These bear correspondence in OWL with the so-called *datatype properties*, which are used to represent attributes or intrinsic properties of a concept. What characterizes datatype properties, from a strictly formal point of view, is that they associate a class (implemented as a domain) to a literal, which is a XML Schema Datatype (XSD) value with which we can represent character strings, integers, etc. In the following Figure we can observe the form we used to implement the property *dateReturn* (date of return), with its respective labels, and with the indication that its domain is the class *meansTransport* (means of transport) and its range, a *xsd:string* literal. This means that the property *dateReturn* applies to the class *meansTransport* and that its values can take the form of strings.



The screenshot shows a web-based property editor interface. At the top, there are two tabs: 'viajeCombinado.ttl' and 'OntologiaVC.rdf'. The main title is 'Datatype Property Form'. Below the title, the 'Name' field contains 'dateReturn'. Under the 'Annotations' section, there is an 'rdfs:comment' field with a text input box. Below that, the 'rdfs:label' section lists four labels with corresponding language flags: 'Rückreisedatum {@de}', 'Tag der Rückkehr {@de}', 'date of return {@en}', and 'fecha de vuelta {@es}'. The 'Property Axioms' section includes 'rdfs:domain' set to 'meansTransport' and 'rdfs:range' set to 'xsd:string'. At the bottom, there are two tabs: 'Form' and 'Source Code'.

Figure 11 Property editor.

The last element that we have included in the ontology at this stage is the definition of each one of the concepts, in Spanish. To that aim, we have resorted to several sources: in the first place, the conceptual information offered by the ontology itself, so, for each definition, we have taken into account the superordinate concept or hypernym, in order to place it as the first word of the definition, whenever it has been possible. In addition, we have made use of the information provided by the relationships and properties.

In the second place, with the purpose of completing the definitions with other data not reflected in the conceptual structure, we have resorted to some general dictionaries, such as the *Diccionario de la Real Academia Española* (Dictionary of the Royal Academy of the Spanish

language) (2001), María Moliner's *Diccionario de uso del español* (Dictionary of Usage of the Spanish language) (2007) and the website of *Oxford Dictionaries* (2010).

Thirdly, some definitions are based on what is set out, defined or explained in the European Directive on package travel (Council Directive 90/314/EEC). Finally, other definitions contain information from specialized dictionaries and webs. The definitions have been implemented through the annotation property `rdfs:comment`. The following Figure illustrates the definition of the concept `retailer`:

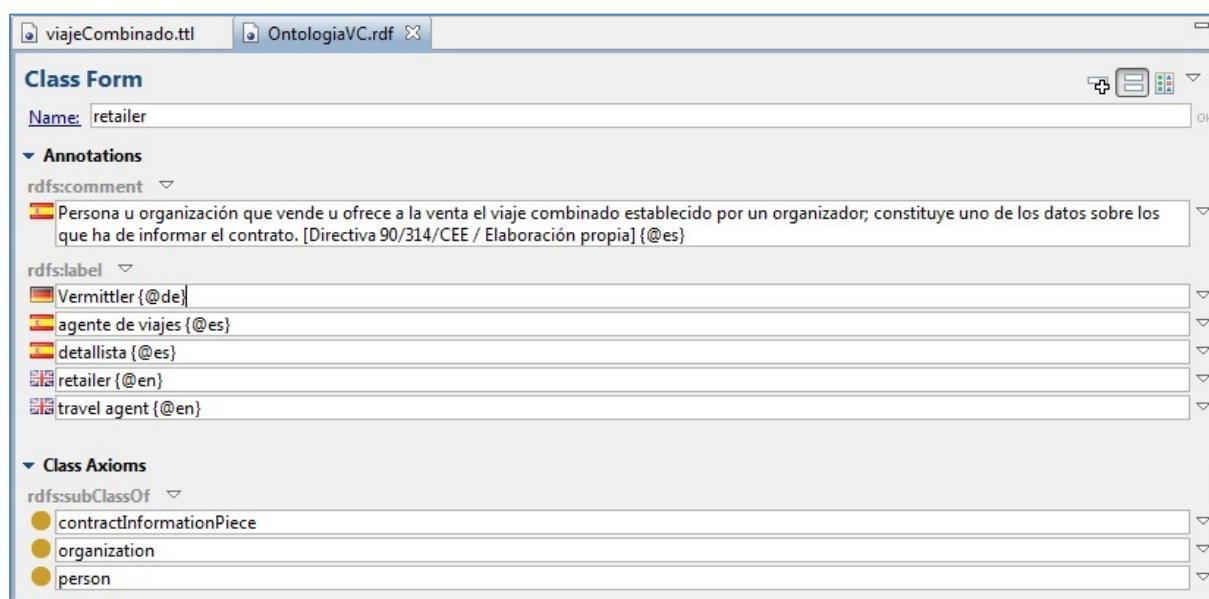


Figure 12 *Definition of retailer.*

In the end we have created an ontology containing 108 concepts, 13 types of relationships (that link in turn many more concepts) and 11 types of properties. We have provided the concepts, as well as the relationships and properties, with linguistic realizations, i.e. terms, in Spanish, English and German.

3 Discussion and conclusions

After constructing our ontology, we have encountered some positive results, as well as some obstacles. With respect to the aspects we established at the beginning, we have obtained the following results:

- Representation of conceptual relationships and characteristics. We have found that constructing an ontology with a standard OWL ontology editor allows representing all types of conceptual relationships in a formal way: both hierarchical (generic and partitive) and associative relations. Characteristics can also be represented by means of datatype properties in OWL.



- Representation of linguistic relationships (synonymy, near synonymy, collocations). We were able to represent paradigmatic relationships such as synonymy by means of the label capabilities of OWL. In this way, if a concept has two possible lexicalizations in a given language, we can add both terms as separate labels with the indication of the corresponding language (for instance, see *retailer* and *travel agent* in Figure 14).
- However, this label functionality does not permit to express near synonymy, nor preferred or admitted terms. As for syntagmatic relationships, such as collocations, we have not found a way to represent them; however, the lexicalizations of the conceptual relationships that we have established can be considered in many cases as collocations as well.
- Use of abstract or *artificial* concepts to better structure the domain. We had to create some classes for classification purposes (cf. L'Homme and Bernier-Colborne, 2012). For example, we had to implement the class `brochureInformationPiece` to encompass all the information items that a brochure must contain. However, we consider that these abstract classes can help solve some categorization divergences that may exist among the conceptual structures of different languages.
- Data categories that can be represented. We have been able to represent several data categories that are standard or widely used in terminography (cf. Cabré, 1999: 124), namely term identifier (by means of the class name), entry term in three different languages (with language identification), definition and its source,²¹ and synonymous terms. However, one major drawback is the lack of flexibility to add different terminological data categories that are normally included in standard terminological records (cf. Cabré, 1999: 124-125): grammatical category, context, or administrative data such as author of the record. Moreover, as we mentioned above, we cannot represent certain linguistic categories such as near synonymy or collocations.
- Ontology display. Since our objective is to provide terminologists and translators with an ontoterminological data bank, we consider that we must try to offer the data stored in the ontology in a user-friendly way. *TopBraid* (or for that matter, any standard ontology editor) is not appropriate for users not familiar with ontologies, but its export capabilities can prove useful: the resulting ontology can be exported to some interchangeable formats and can be thus be viewed by means of ontology browsers such as *OWLViewer*,²² *Ontology Browser*²³ or *SWOOP*,²⁴ although these might pose slight viewing problems. Another option is the use of *OntoDiccionario* (Bautista Zambrana, 2013), an application capable of interpreting the ontology built with *TopBraid* and of displaying it in a user-friendly way, specially aimed at terminologists and translators.

This study set out to determine which possibilities or obstacles we encounter when constructing an OWL ontology for terminological purposes. The results of this study indicate that ontologies are a good instrument to represent conceptual relationships and characteristics in a formal and consistent way, when we wish to build an ontoterminological data bank or dictionary. Moreover, ontologies offer the possibility of expressing that knowledge in several languages, as we have observed with the use of labels. This is without a doubt interesting, all the more so

²¹ We have added a definition in Spanish for each concept, but we could have done it as well for English and German.

²² http://agem.cnb.csic.es/VisualOmic/OwlViewer/index_OV.html

²³ <http://code.google.com/p/ontology-browser>

²⁴ <http://code.google.com/p/swoop>

because of the current research being made in the Linked Open Data framework, which seeks to expose, share and connect pieces of data, information and knowledge on the Semantic Web (Declerck and Wandl-Vogt, 2014). As these authors state, “by interlinking multilingual and open (language) resources, we foresee a linguistic linked open data (LLOD) cloud, a new linguistic ecosystem, that will allow the open exploitation of such data.” Terminology could undoubtedly benefit from that initiative.

However, when constructing our ontology, we had to compromise on some other aspects, as L’Homme and Bernier-Colborne (2012) point out. We were not able to represent certain linguistic relationships such as near synonymy, term preference, or collocations. We could not include textual contexts or usage notes either. On the other hand, although creating abstract concepts facilitates the representation of the conceptual structure of a domain across languages and cultures, it still remains difficult to organize the conceptual model in a way that can accommodate all the categorization differences that can arise when dealing with culturally-dependent domains, such as tourism legislation.

We are aware that external linguistic models such as Lemon address and solve some of these drawbacks (for instance, term preference), but we consider that further research is needed to improve the representation of terminologies by means of ontologies, or to build a better knowledge-based approach to terminology.

Acknowledgements

The research reported in this paper has been carried out in the framework of the R&D projects “INTELITERM: Sistema inteligente de gestión terminológica para traductores” (reference no. FFI2012-38881) and “TERMITUR: Diccionario inteligente TERMIInológico para el sector TURístico (alemán-inglés-español)” (reference no. HUM2754, 2014-2017. Junta de Andalucía).

Bibliography

- Bautista Zambrana, M. R. (2013): *Diseño y elaboración de un glosario ontoterminológico multilingüe para el viaje combinado (español-inglés-alemán)*. Doctoral thesis, University of Málaga (Spain).
- Beale, S., Nirenburg, S. & Mahesh, K. (1995): Semantic Analysis in the Mikrokosmos Machine Translation Project. In *Proceedings of the Second Symposium on Natural Language Processing (SNLP-95)*. Kaser Sart University, Bangkok, Thailand.
- Cabré Castellví, M. T. (1999): *Terminology: theory, methods and applications*. John Benjamins: Amsterdam; Philadelphia.
- Cabré Castellví, M. T. (2004): Los bancos de conocimiento especializado multilingüe: un nuevo recurso para la traducción. In *Actas del Congreso El español, lengua de traducción. II Congreso Internacional*. Toledo (Spain). <http://www.esletra.org/Toledo/html/contribuciones/cabre.htm>
- Clark, K., Parsia, B. & Hendler, J. (2004): Will the Semantic Web Change Education? *Journal of Interactive Media in Education. Special Issue on the Educational Semantic Web*, 3. <http://www-jime.open.ac.uk/2004/3/clark-2004-3-disc-05.html>
- Corcho, Ó., Fernández-López, M., Gómez-Pérez, A. & López-Cima, A. (2005): Construcción de ontologías legales con la metodología METHONTOLOGY y la herramienta WebODE. In *Law and the Semantic Web. Legal Ontologies, Methodologies, Legal Information Retrieval, and Applications* (pp. 142-157). Springer-Verlag.
- Declerck, T. & Wandl-Vogt, E. (2014): Publishing and consuming lexicographical resources in the linked (open) data cloud. http://euralex2014.eurac.edu/de/programme/Documents/Lexicographicalres_link_ed_data_cloud_def.pdf
- Durán-Muñoz, I. & Bautista-Zambrana, M. R. (2013): Applying Ontologies to Terminology: Advantages and Some Disadvantages. *Hermes - Journal of Language and Communication in Business*, 51: 65-77.
- Faber, P., León Araúz, P. & Prieto-Velasco, J. A. (2009): Semantic Relations, Dynamicity, and Terminological Knowledge Bases. *Current Issues in Language Studies*, 1: 1-23.
- Feliu, J.; Vivaldi, J.; Cabré Castellví, M. T. (2002): *Ontologies: A Review*. <https://repositori.upf.edu/bitstream/handle/10230/1295/02inf034.pdf>
- Fernández-López M., Gómez-Pérez A. & Juristo N. (1997): METHONTOLOGY: From Ontological Art Towards Ontological Engineering. In *Spring Symposium on Ontological Engineering of AAAI* (pp. 33-40). Stanford University, California.
- Genesereth, M. R. & Nilsson, N. J. (1987): *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers: San Mateo, California.
- Gómez Pérez, A., Juristo, N., Montes, C. & Pazos, J. (1997): *Ingeniería del conocimiento*. Centro de Estudios Ramón Areces: Madrid.
- Gómez-Pérez, A., Fernández-López, M. & Corcho, Ó. (2004): *Ontological Engineering: with examples from the areas of Knowledge Management*. Springer-Verlag: London.
- Grabar, N., Hamon, T. & Bodenreider, O. (2012): Ontologies and terminologies: Continuum or dichotomy? *Applied Ontology*, 7: 375-386.
- Gromann, D. (2013): Terminology meets the multilingual Semantic Web: A semiotic comparison of ontologies and terminologies. In *Proceedings of the 19th European Symposium on Languages for Special Purposes*. University of Vienna, Vienna. <https://fedora.phaidra.univie.ac.at/fedora/get/o:359149/bdef:Content/get>

- Gruber, T. (1993): A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2): 199-220.
- Hirst, G. (2009): Ontology and the Lexicon. In S. Staab & R. Studer (eds.): *Handbook on Ontologies* (pp. 269–292). Springer: Berlin. <http://ftp.cs.toronto.edu/pub/gh/Hirst-Ontol-2009.pdf>
- Leonardi, N. (2012): ‘Ontology’ and Terminological Frameworks: an Overview of Issues and Term(s). *Hermes - Journal of Language and Communication in Business*, 48(3): 19-33.
- L'Homme, M.-C. & Bernier-Colborne, G. (2012): Terms as labels for concepts, terms as lexical units: A comparative analysis in ontologies and specialized dictionaries. *Applied Ontology*, 7(4): 387–400.
- Moliner, M. (2007): *Diccionario de uso del español*. Editorial Gredos: Madrid.
- Montiel, E. (2011): *Multilingualism in Ontologies - Multilingual Lexico-Syntactic Patterns for Ontology Modeling and Linguistic Information Repository for Ontology Localization*. Doctoral thesis, Universidad Politécnica de Madrid (Spain).
- Moreno Ortiz, A. (2008): Ontologías para la Terminología: Por Qué, Cuándo, Cómo. *Tradumàtica*, 6. <http://www.fti.uab.cat/tradumatica/revista/num6/articles/03/03central.htm>
- Øhrstrøm, P., Andersen, J. & Schärfe, H. (2005): What has happened to ontology. In F. Dau, M. L. Mugnier & G. Stumme (eds.): *Proceedings of the Conceptual Structures: Common Semantics for Sharing Knowledge, 13th International Conference on Conceptual Structures, ICCS 2005* (pp. 425-438). Springer: Heidelberg.
- Oxford Dictionaries* (2010). Oxford University Press. <http://oxforddictionaries.com>
- Real Academia Española (2001): *Diccionario de la Lengua Española* (DRAE). 22th ed. Espasa-Calpe: Madrid. <http://rae.es/rae.html>
- Sowa, J. (1999): Signs, Processes, and Language Games: Foundations for Ontology (extended version of an invited lecture presented at the International Conference on the Challenge of Pragmatic Process Philosophy at the University of Nijmegen in May 1999). <http://www.jfsowa.com/pubs/signproc.htm>
- Summers, D. (dir.) (1995): *Longman Dictionary of Contemporary English*, 3rd ed. Pearson Education Limited: Harlow.
- TopQuadrant (2011): *TopBraid Composer. Getting Started Guide. Version 3.0*. <http://www.topquadrant.com/docs/marcom/TBCGetting-Started-Guide.pdf>
- Temmerman, R. & Kerremans, K. (2003): Termonotography: Ontology Building and the Sociocognitive Approach to Terminology Description. In E. Hajicová, A. Kotešovicová & J. Mírovský (eds.): *Proceedings of CIL17* (pp. 1-10). Matfyzpress, MFF UK: Prag.
- Valeontis, K. & Mantzari, E. (2006): The linguistic dimension of terminology: Principles and methods of term formation. In *1st Athens International Conference on Translation and Interpretation Translation: Between Art and Social Science*. http://www.eleto.gr/download/BooksAndArticles/HAU-Conference2006-ValeontisMantzari_EN.pdf