# Chapter 4
# The NeOn Ontology Models

**Alessandro Adamou, Raúl Palma, Peter Haase, Elena Montiel-Ponsoda, Guadalupe Aguado de Cea, Asunción Gómez-Pérez, Wim Peters, and Aldo Gangemi**

**Abstract** Interoperability on multiple levels, concerning both the ontologies themselves and their engineering activities, is a key requirement for ontology networks to be efficient, with minimal redundancy and high reuse. This requirement has a strict binding for software tools that can support some interoperability levels, yet they can be hindered by a lack of shared models and vocabularies describing the resources to be handled, as well as the ways of handling them. Here, three examples of metalevel vocabularies are proposed, each covering at least one peculiar

A. Adamou (✉)
Semantic Technologies Lab, Institute of Cognitive Sciences, and Technologies (National Research Council – CNR), Via Nomentana 56, 00161 Rome, Italy

Department of Computer Science, Alma Mater Studiorum Universitá di Bologna, Mura Anteo Zamboni 7, 40126 Bologna, Italy
e-mail: alessandro.adamou@istc.cnr.it; adamou@cs.unibo.it

R. Palma
Poznan Supercomputing and Networking Center, ul. Dabrowskiego 79a, 60-529 Poznan, Poland
e-mail: rpalma@man.poznan.pl

P. Haase
fluid Operations AG, Altrottstr. 31, 69190 Walldorf, Germany
e-mail: peter.haase@fluidops.com

E. Montiel-Ponsoda • G. Aguado de Cea • A. Gómez-Pérez
Ontology Engineering Group, Facultad de Informática, Universidad Politécnica de Madrid, Campus de Montegancedo sn, 28660 Boadilla del Monte, Madrid, Spain
e-mail: emontiel@fi.upm.es; lupe@fi.upm.es; asun@fi.upm.es

W. Peters
University of Sheffield, Sheffield, UK
e-mail: w.peters@dcs.shef.ac.uk

A. Gangemi
Semantic Technologies Lab, Institute of Cognitive Sciences, and Technologies (National Research Council – CNR), Via Nomentana 56, 00161 Rome, Italy
e-mail: aldo.gangemi@cnr.it

interoperability aspect: OMV for modeling the artifacts themselves, LIR for managing a multilingual layer on top of them, and C-ODO Light for modeling collaboration-supportive life cycle management tasks and processes. All of these models lend themselves to handling by dedicated software tools and are all being employed within NeOn products.

## 4.1 Introduction

Authoring ontologies and modeling domains of interest are only part of an ontology life cycle management process. If these activities are carried out in a monolithic fashion, from scratch and without reusing readily available knowledge models, this may lead to costly "reinventions of the wheel" and contradicts the Semantic Web philosophy of an open knowledge world. On the other hand, even when the intention and sentiment to follow this philosophy are present, they might not be encouraged by appropriate tool support. This, in turn, depends on the availability of formal models of processes and artifacts in ontology design, i.e., their *metalevel*. This model may sometimes be implicitly hardwired in the software itself, but if it is not, then it may be useful to share and exploit it for the sake of interoperability, be it conceptual, linguistic, functional, or social.

This chapter focuses on three contributions to the practice of ontology design by metalevel handling. Each contribution, presented itself as an ontology network, covers a specific design perspective, i.e., reuse (OMV), localization (LIR), and collaborative engineering (C-ODO Light). By the end of the chapter, the reader will have a practical insight as to how a model of the ontology metalevel can be employed to build effective software tools to automate engineering tasks.

## 4.2 Ontology Metadata Vocabulary (OMV)

Ontologies have undergone an enormous development and have been applied in many domains within the last years, especially in the context of the Semantic Web. Currently, however, efficient knowledge sharing and reuse, a prerequisite for the realization of the Semantic Web vision, is a difficult task. It is hard to find and share existing ontologies because of the lack of standards for documenting and annotating ontologies with metadata information. Without ontology-specific metadata, developers are not able to reuse existing ontologies, which leads to interoperability problems, as well as duplicate efforts. In order to provide a basis for an effective access and exchange of ontologies across the web, it is necessary to agree on a standard for ontology metadata. This standard then provides a common set of terms and definitions describing ontologies and is called *metadata vocabulary*.

*Limitations*. The need for a metadata vocabulary for describing ontologies has been acknowledged in the past by previous efforts (e.g., Dublin Core 1998, Reference Ontology 2000, Ontology Metaontology (OMO) 2003, and DogmaModeler Ontology 2005). However, at the moment, most of the current ontologies exist in pure form without any additional information, e.g., domain of interest, authorship information, and statistic information (Ungrangsi and Simperl 2008). This is due in part to the *lack of standards or community-accepted vocabularies* for documenting and annotating ontologies with metadata information. Moreover, most of the previous efforts carried out on this issue provide only a *list of property-value pairs* for describing ontologies (e.g., Arpírez et al. 2000; Jarrar 2005), limiting the processing capabilities and the related relevant information that can be described. Similarly, ontology metadata are in many of the existing systems and repositories (e.g., DAML Ontology Library[1], SchemaWeb Directory[2], and SWOOGLE[3]), not based on agreed standards, which makes them difficult to integrate or reuse. Finally, *general-purpose standards*, such as Dublin Core, *are not appropriate* for capturing information about ontologies because of the differences between arbitrary information sources and ontologies. For instance, aspects related to the application scenario, scope, purpose, or evaluation results are essential when describing ontologies. Additionally, besides structural and technical information, ontologies have to be described in terms of descriptive metadata, such as provenance information, ontology categorizations, underlying methodologies, or knowledge representation paradigms that are specific for ontologies.

Thereupon, in this chapter we describe our contribution to the alleviation of this situation: the ontology metadata standard OMV (Ontology Metadata Vocabulary), which specifies reusability-enhancing ontology features for human- and machine-processing purposes. It allows to clarify the relations between the available ontologies so that they are easy to search, to characterize, and to maintain. Moreover, it provides the means for making explicit the virtual and implicit network of ontologies.

*Ontology Metadata Requirements*. As a result of a systematic survey of the state of the art in the area of ontology reuse, we have elaborated an inventory of requirements for the metadata model. Besides analytical activities, we conducted extensive literature research focused on theoretical methods (Pinto and Martins 2001; Gangemi et al. 1999; Lozano-Tello and Gómez-Pérez 2004) and also on case studies on reusing existing ontologies (Uschold et al. 1998; Russ et al. 1999; Paslaru Bontas et al. 2005). Our aim was to identify the real-world needs of the community with respect to a descriptive metadata format for ontologies. Further on, the requirement analysis phase was complemented by a comparative study of existing (ontology-independent) metadata models and tools such as ontology repositories and libraries that (implicitly) make use of some form of ontology metadata.

[1] http://www.daml.org/ontologies/

[2] http://www.schemaweb.info/

[3] http://swoogle.umbc.edu/

Several aspects to be considered in ontology metadata representation are definitely similar to those of other more general metadata standards such as Dublin Core. Differences arise, however, if we consider the semantic nature of ontologies, which are much more than plain web information sources. The main requirements identified in this process are the following:

*Accessibility*. Metadata should be accessible and processable for both humans and machines. Whereas the human-driven aspects are ensured by the usage of natural language concept names, the machine-readability requirement can be implemented by the usage of web-compatible representation languages (such as XML or Semantic Web languages, see below). Furthermore, having metadata in processable format will facilitate the implementation of tools that use or manage ontology-related metadata (e.g., ontology changes).

*Usability*. A metadata model should (1) reflect the needs of the majority of ontology users, as reported by existing case studies in ontology reuse, but at the same time (2) allow proprietary extensions and refinements in particular application scenarios (e. g., ontology change management). From a content perspective, usability can be maximized by taking into account multiple metadata types, which correspond to specific viewpoints on the ontological resources and are applied in various application tasks. Despite the broad understanding of the metadata concept and the use cases associated to each definition, several key aspects of metadata information have already been established across computer science fields (NISO 2004):

- *Structural metadata* relate to statistical measures on the graph structure underlying an ontology. In particular, we mention the number of specific ontological primitives (e.g., number of classes and individuals). The availability of structural metadata influences the usability of an ontology in concrete application scenarios, because size and structure parameters constrain the type of tools and methods that are applied to aiding the reuse process. For instance, as it has been analyzed in the past (e.g., Gardiner et al. 2006), most ontology reasoners have still scalability issues when dealing with large ontologies. Furthermore, structural metadata provide core information to identify when an ontology changes (e.g., a different number of classes or individuals).
- *Descriptive metadata* relate to the domain modeled in the ontology in the form of keywords, topic classifications, textual descriptions of the ontology contents, etc. This type of metadata plays a crucial role in the selection of appropriate reuse candidates, a process that includes requirements with respect to the domain of the ontologies to be reused. Moreover, descriptive metadata are highly useful when identifying ontology changes from a high-level point of view (e.g., the domain has been specialized and the description has been updated).
- *Administrative metadata* provide information to help manage an ontology, such as when and how it was created, rights management, file format, and other technical information. Obviously, information like the date of modification of the ontology is also useful to identify when an ontology has changed.

*Interoperability*. Similar to the ontology it describes, metadata information should be available in a form that facilitates metadata exchange among applications. While the syntactical aspects of interoperability are covered by the usage of standard representation languages (see ‘Accessibility’), the semantic interoperability among machines handling ontology metadata information can be ensured by means of a formal and explicit representation of the meaning of the metadata entities (by conceptualizing the metadata vocabulary itself as an ontology).

## *4.2.1 OMV Overview*

This section presents the ontology metadata vocabulary (OMV); the first part provides an overview of the core design principles applied to the development of the OMV metadata model; then, we describe in detail the core of such a model; next, we present implementation and practical aspects; finally, we provide an introduction to the OMV extensions.

### 4.2.1.1 Core and Extensions

Following the usability constraints identified during the requirements analysis, we decided to design the OMV schema modularly, distinguishing between the OMV core and various OMV extensions. The former captures information that is expected to be relevant to the majority of ontology reuse settings. However, in order to allow ontology developers and users to specify task-or application-specific ontology-related information, we allowed for the development of OMV extension modules, which are separated from the core schema while remaining compatible to it. That is, the terms are supposed to mean the same thing in the core and the extensions. Essentially, extensions reuse the core knowledge and provide specialized information for different ontology aspects.

### 4.2.1.2 Metadata Organization and Categorization

In the following, we present the organization and categorization of metadata (entities) in two dimensions, which provide a structured overview of the OMV ontology:

*Property Appropriation*. We organize metadata entities according to the impact on the prospected reusability of the described ontological content as presented in the following list:

- Required – mandatory metadata elements. Any missing entry in this category leads to an incomplete description of the ontology.
- Optional – important metadata facts, but not strongly required.

- Extensional – specialized metadata entities, which are not considered to be part of the core metadata schema.

*Property Categorization*. Orthogonal to the previous classification, we organize the metadata elements according to the type and purpose of the information contained as follows:

- General – elements providing general information about the ontology.
- Availability – information about the location of the ontology (e.g., its URI or the URL where the ontology is published on the web).
- Applicability – information about the intended usage or scope of the ontology.
- Format – information about the physical representation of the resource. In terms of ontologies, these elements include information about the representation language(s) in which the ontology is formalized.
- Provenance – information about the organizations contributing to the creation of the ontology.
- Relationship – information about relationships to other resources. This category includes versioning, as well as conceptual relationships such as extensions, generalization/specialization, and imports.
- Statistics – various metrics on the underlying graph topology of an ontology (e.g., number of classes).
- Other – information not covered in the categories listed above.

Note that the classification dimensions introduced above (appropriation and categorization) are intended to be considered when implementing several metadata support facilities. The first dimension is relevant for a metadata creation service, since it ensures a minimal set of useful metadata entries for each of the described resources. The second can be used in various settings, mainly to reduce the user-perceived complexity of the metadata schema, whose elements can be structured according to the corresponding categories.

#### 4.2.1.3 OMV Core Metadata Entities

The main classes and properties of the OMV ontology are illustrated in Fig. 4.1[4].

Besides the main class Ontology, the metadata model contains elements describing various aspects related to the creation, management, and usage of an ontology. We will briefly discuss these in the following text. In a typical ontology engineering process, person(s) or organization(s) develop ontologies. We group these two classes under the generic class Party by a subclass-of relation. A Party can have several locations by referring to a Location individual and can create and contribute

[4] Please notice that not all classes and properties are included. The ontology is available for download in several ontology formats at http://omv.ontoware.org/

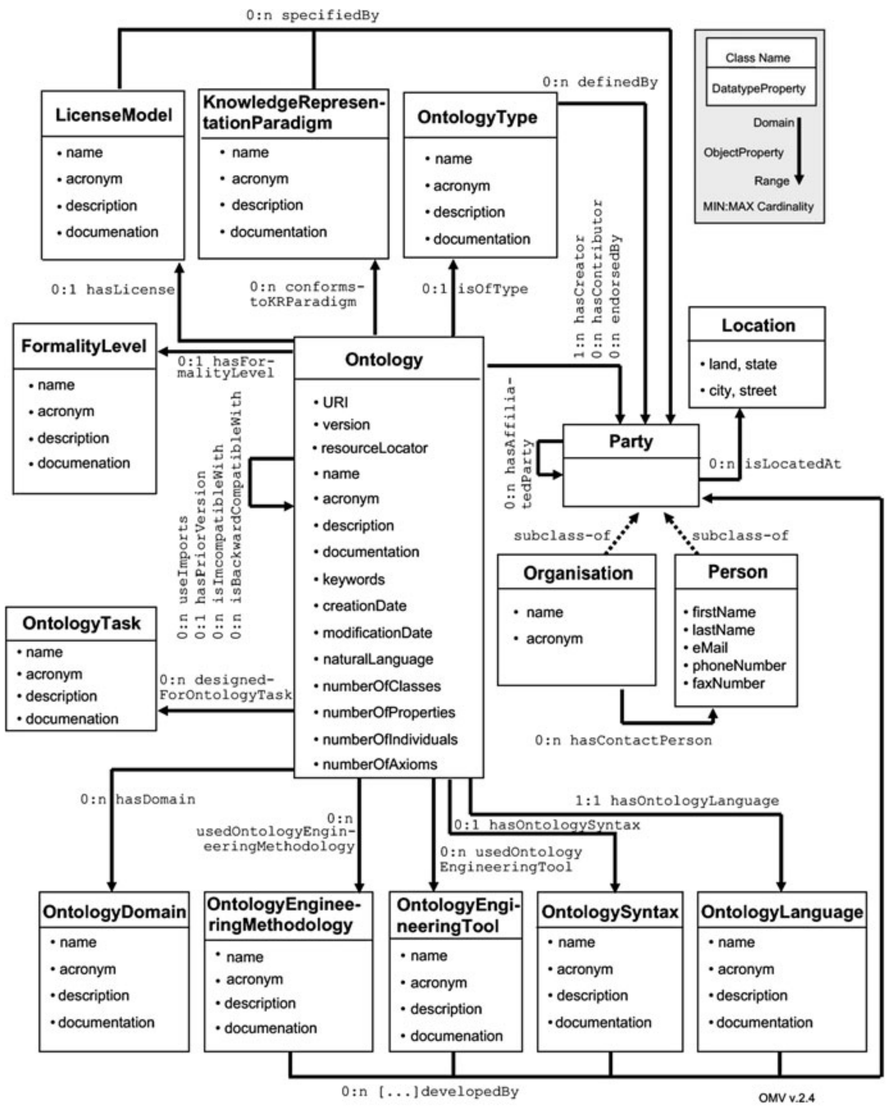

**Fig. 4.1** OMV core overview

to ontological resources, i.e., Ontology class. Review details and further information can be captured in an extensional OMV module. Further on we provide information about the engineering process the ontology originally resulted from in terms of the classes `OntologyEngineeringMethodology`, `OntologyEngineeringTool`, and the attributes `version`, `status`, `creationDate`, and `modificationDate`. Again these can be elaborated as an extension of the core metadata schema. The usage history of the ontology is modeled by classes such as the OntologyTask and LicenceModel. The scheme also contains a representation of the most significant intrinsic features of an ontology. Details on ontology languages are representable with the help of the classes OntologySyntax, OntologyLanguage, and KnowledgeRepresentationParadigm. Ontologies might be categorized along a multitude of dimensions. One of the most popular classifications differentiates among application, domain, core, task, and upper-level ontologies. A further classification relies on their level of formality and types of Knowledge Representation (KR) primitives supported, introducing catalogs, glossaries, thesauri, taxonomies, frames, etc., as types of ontologies. The former categories can be modeled as individuals of the class OntologyType, while generic formality levels are introduced with the help of the class FormalityLevel. The domain the ontology describes is represented by the class OntologyDomain that references a predefined topic hierarchy such as the DMOZ hierarchy. Further content information can be provided as values of the attributes description, keywords, and documentation. Moreover, the metadata schema provides information about the imported ontologies (useImports) and versioning relations (hasPriorVersion, isBackwardCompatibleWith, and isIncompatibleWith) – analogously to the OWL ontology properties. Finally, OMV gives an overview of the graph topology of an Ontology with the help of several graph-related metrics represented as integer values of the attributes numberOfClasses, numberOfProperties, numberOfAxioms, and numberOfIndividuals.

#### 4.2.1.4 Ontological Representation

Following the accessibility and interoperability requirements, as well as the nature of the metadata, which are intended to describe ontologies, the conceptual model designed in the previous steps was implemented in OWL2[5]. With OWL being established as the standard to represent ontologies, it was only logical to opt for representing ontology metadata using the same language. As a consequence, the same tooling for processing the ontologies can also be used for processing the ontology metadata.

Additionally, a metadata element is modeled either by means of classes and individuals or by means of valued properties. The former alternative, represented using additional classes linked by object properties, was chosen to model those metadata elements representing entities that can be referred to. The latter alternative, represented using datatype properties, was chosen to model metadata elements with value/content that can be easily mapped to conventional data types (numerical, literal, list values).

[5] In the remainder of this chapter, when OWL appears without any version information, it refers to OWL1. As opposed, when referring to OWL2, we explicitly note it.

Finally, OMV implements the appropriate properties of metadata entities by different means: The required and optional metadata entities are implemented in OMV core with the appropriate cardinality restrictions, while the extensional metadata entities are implemented in the different OMV extensions.

#### 4.2.1.5 OMV Extensions

The OMV core metadata is intended to evolve toward a commonly agreed schema for Semantic Web ontologies. In contrast to this ambitious goal, we are aware that for specific domains, tasks, or communities, extensions in any direction might be required. These extensions should be compatible to the OMV core, but at the same time, they should fulfill the requirements of a domain-, task-, or community-driven setting.

The character of an OMV extension is a metadata ontology itself that imports the OMV core ontology. There are no restricting modeling guidelines to be met. However, developers are encouraged to follow the design principles described above (see Sects. 4.2.1.2 and 4.2.1.3), as well as to follow a basic set of guidelines for naming ontology terms (Palma et al. 2008).

Some of the existing OMV extensions were developed in collaboration with different institutions. The available extensions[6] are: the generic change ontology, which models changes to an ontology (Palma 2009); the lexOMV extension (Montiel-Ponsoda et al. 2007) that models the linguistic or multilingual data contained in the ontology; the modules extension that represents the description of ontology modules (d'Aquin et al. 2008); the peer extension that captures information of peers sharing metadata about ontologies and related entities (e.g., mappings and modules) (Wang et al. 2007); and the mapping extension that describes mappings between heterogeneous ontologies.

### *4.2.2 Uses and Benefits*

OMV plays an important role in the ontology reuse task by facilitating the discovery and exchange of ontologies, fostering the widespread dissemination of ontology-driven technologies and the development of full-fledged ontology repositories and registries on the web. Furthermore, applications that work with the creation or (re)use of ontologies can benefit from having a standard schema for ontology metadata. By using the same vocabulary to describe ontology metadata, applications can exchange this information easily.

[6] OMV extensions are also available at http://omv.ontoware.org

Several applications are already using OMV to describe ontology metadata. In this section, we present a selection of these applications that use OMV at various stages of the ontology development life cycle. First, the NeOn Toolkit[7] (c.f. Part III of this book) includes a set of OMV-related plugins that either use OMV or provide access to OMV-enabled registries (e.g., Oyster, Centrasite). Oyster[8] is an open-source ontology registry that uses the metadata for retrieval and selection tasks and can also export OMV data for other applications. Similarly, the commercial registry Centrasite[9] provides an OMV-specialized web service to support the management of ontology metadata. Also, BioPortal[10] and Cupboard[11] are two ontology repositories that use OMV for the description of ontologies. The Semantic Web gateway Watson[12] can generate OMV annotations for the ontologies discovered. Finally, applications such as the Protege MetaAnalysis plugin[13] allow to calculate various metadata for ontologies and facilitate the export of that metadata to the OMV.

*Oyster* (Palma and Haase 2005) is a distributed registry that exploits Semantic Web techniques in order to provide a solution for exchanging and reusing ontologies and related entities (e.g., ontology developers, ontology mappings, ontology changes, etc.). To achieve this goal, Oyster uses OMV to describe ontologies and related entities.

Moreover, Oyster uses ontologies extensively to provide its main metadata management functions (registry metadata, formulating queries, routing queries, and processing answers). The ontology metadata entries are aligned and formally represented according to two ontologies: (1) the OMV that describes the properties of the ontology and (2) a topic hierarchy to define the domain of the ontology (c.f. Sect. 4.3).

*NeOn Applications*. The NeOn Toolkit includes different OMV-related plugins. The Oyster-API plugin enables programmatic access to all Oyster registry functionalities within any other NeOn Toolkit plugin. This plugin can either use a local or remote Oyster instance. Similarly, the Oyster-GUI plugin provides a graphical user interface to interact with Oyster servers and other OMV-enabled servers (e.g., Centrasite) implementing the OMV-based web service. This plugin allows submitting, updating, and removing instances of OMV core classes, submitting queries to search ontologies based on different criteria and importing from the Internet ontologies matching the search criteria. Furthermore, the change-capturing plugin implements methods and strategies for the capturing and synchronization of ontology changes that are formally represented as instances of the change ontology (an OMV

[7] http://www.neon-toolkit.org/

[8] http://oyster2.ontoware.org

[9] http://www.infoq.com/zones/centrasite/

[10] http://bioportal.bioontology.org

[11] http://cupboard.open.ac.uk:8081/cupboard

[12] http://watson.kmi.open.ac.uk/

[13] http://protegewiki.stanford.edu/wiki/MetaAnalysis

extension). This OMV extension is also used by several other plugins, such as Cicero plugin (to enable discussions on changes), Evolva plugin (to represent the changes proposed by the plugin based on background knowledge) and GATE Web service plugin (to represent the changes generated from textual sources).

Additionally, the Cupboard system produced in NeOn for ontology publishing, sharing, and reuse also relies on OMV to implement some of its features. Besides letting users add their ontologies in a personal space – hosting, indexing, linking, and exposing them through APIs and SPARQL – Cupboard is designed to be a community tool. It helps ontology users and practitioners (including ontology developers) in finding and reusing ontologies, through the use of rich ontology metadata (thanks to Oyster and OMV) and advanced ontology review mechanisms.

Finally, the latest update produced in NeOn of the collaborative ontology design ontology (C-ODO), called *codolight* (c.f. Sect. 4.4), has been aligned with OMV. Compared to the original C-ODO ontology design metamodel, *codolight* is now linked to requirements and application tasks, has been used for tool descriptions, is aligned to external vocabularies, is lighter in complexity, and improves association between the social and software layers of ontology design aspects. From a design viewpoint, the metadata provided by OMV have a semantics that is potentially compatible to that of other metamodels, and this alignment helps with metadata interoperability.

*Protege Plugin*. The Protege MetaAnalysis plugin calculates various metadata for ontologies and facilitates the export of those metadata to the OMV. The plugin is a tab widget consisting of four panels: the numbers panel, the design panel, the OWL panel, and the extras panel. The plugin computes metadata for a given ontology and displays them in these panels. The ontology metadata can be exported to an extension of the OMV. If the ontology already exists in OMV, the metadata for that ontology are updated. Otherwise, a new instance of the ontology is created in OMV and populated with the computed metadata.

*BioPortal*. While Oyster is a distributed ontology repository, BioPortal is a centralized repository of biomedical ontologies, where authors submit their ontologies. As part of the submission process, authors also fill in the form to describe their metadata. In the future, it is planned to add the capability for the authors simply to point to the location of an OWL file that has the OMV individuals and to have BioPortal import the information from that file.

BioPortal uses the ontology metadata in ontology search and navigation. Users can specify, e.g., whether they want their search term to appear only in concept definitions or in metadata as well. BioPortal will also use OMV extensions. For example, one of the functions of BioPortal is to be a repository of mappings between concepts in biomedical ontologies. Each mapping comes with its own set of metadata (e.g., the mapping author, the algorithms used, the application context in which the mapping is valid, etc.) (Fridman Noy et al. 2008). It is planned to represent the mapping metadata as an OMV extension.

Another feature of BioPortal is the use of peer reviews for ontology evaluation. Ontology users can rate BioPortal ontologies along different dimensions, such as coverage and degree of formality, based on their experience with the ontology in their own applications (Fridman Noy et al. 2005). The evaluation extension will also be an OMV extension.

The *Watson Semantic Web gateway* contains a repository of ontologies and provides export of their metadata in the OMV format. When Watson users search for ontologies, they can click on an ontology URI from the search results, and then on the overview page for that, click on "Get OMV" for the metadata export.

*OMEGA* is an algorithm that addresses the problem of populating metadata elements (Ungrangsi and Simperl 2008). It generates automatically metadata about arbitrary ontologies on the web and is available as a web application and a REST web service. It takes an ontology as input and automatically populates certain metadata information such as domain, level of formality, and statistics, using an ontology metadata schema, which is part of the OMV standard.

## 4.3 Linguistic Information Repository (LIR)

The symbiosis between ontologies and natural language has proven more and more relevant in the light of the growing interest and use of Semantic Web technologies. Ontologies that are well-documented in a natural language not only provide humans with a better understanding of the world model they represent, but also a better exploitation by the systems that may use them. This "grounding in natural language" is believed to provide improvements in tasks such as ontology-based information extraction, ontology learning, and population from text or ontology verbalization (Buitelaar et al. 2009).

Nowadays, there is a growing demand for ontology-based applications that need to interact with information in different natural languages, i.e., with multilingual information. This is the case of numerous international organizations currently introducing semantic technologies in their information systems, such as the Food and Agriculture Organization or the World Health Organization, to mention just a few. Such organizations have to manage information and resources available in more than a dozen of different natural languages and have to customize the information they produce to a similar number of linguistic communities.

For all these reasons, solutions have to be provided to model multiple natural language descriptions in ontologies. Such an undertaking needs to consider several requirements imposed by the characteristics of the domain of knowledge modeled in the ontology and by the type of linguistic descriptions that are required by the final application.

*Requirements.* Although the number of multilingual ontologies is still quite small compared with the total amount of ontologies available in the web[14], we have conducted a survey of the state of the art of modeling options to represent multilingual information in ontologies (Montiel-Ponsoda et al. 2010). This survey

[14] The Semantic Web search engine Watson provides data about the language of ontology labels that shows that around 80% of ontologies have literals only in English (http://watson.kmi.open.ac.uk/blog/2007/11/20/1195580640000.html)

has revealed the existence of three modeling options, which are briefly explained in the following:

- Including multilingual labels in the ontology model
- Combining the ontology model with a mapping model between different natural languages or a common interlingua
- Associating the ontology model with an external linguistic model

The first modeling option relies on the RDF(S) and OWL properties `rdfs:label` and `rdfs:comment` to associate word forms and descriptions to ontology elements. The main disadvantage of this option is that it is not possible to define any relation among the linguistic annotations, so that the linguistic information is restricted and the model is difficult to scale. The second option assumes the existence of several ontologies in the same domain with labels expressed in different natural languages, which are mapped to each other in a pairwise fashion, or through a common conceptualization or interlingua. This option has been considered in projects such as *EuroWordNet* (Vossen 1998). The risk of this option is that a lot of effort has to be put in developing one conceptualization per language and also in establishing mappings or links among conceptualizations. Finally, the third modeling option allows the association of an external model of linguistic descriptions to the ontology. The main advantages of this modeling option have to do with the capability of the linguistic model of evolving into a complex model of linguistic descriptions that can be accommodated to account for the needs of the final applications. Models that follow this approach include *LingInfo* (Buitelaar et al. 2006), *LexOnto* (Cimiano et al. 2007) or *LexInfo* (Buitelaar et al. 2009).

Whereas some models have been explicitly designed to enrich ontologies with linguistic information, such as the ones mentioned above, they mainly focus on morphosyntactic descriptions of ontological entities and have not handled multilingualism issues, as the ones that arise when aiming at reusing the same ontology in different linguistic and cultural settings. This is particularly relevant in the case of ontologies that represent categorizations of reality that are not completely valid for all the cultures and languages involved. In this context, we have to consider the possibility of providing relations among the linguistic descriptions in different languages associated to the same ontology elements.

Finally, we refer to the need for encoding the linguistic descriptions captured in the linguistic model according to standard models in order to guarantee interoperability, reuse, and commitment to best practices. The potential integration of terminological and lexical knowledge bases into our model requires interoperability with existing and proposed standards. In this sense, we have analyzed some standardization initiatives that have been developed in order to capture linguistic information that can be reused for various purposes. As the most important initiatives, we mention a number of standards from the International Organization for Standardization (ISO) and the World Wide Web Consortium (W3C) that capture terminological and lexical information. We are referring to Terminological Markup Framework (TMF) (ISO 2003), the Lexical Markup Framework (LMF) (ISO 2006), and Simple Knowledge Organization Systems (SKOS) (Miles et al.

2005). After the analysis of the state of the art and considering the needs of a model that aims at providing ontologies with multilingual information, we identified the following set of requirements:

*Independence*: the possibility for providing independent and complex models of linguistic information that can be self-contained and from which information can be inferred. The independence between the ontology and the linguistic model guarantees the full development of both without one restricting the other. In particular, in the case of the linguistic model, this allows the existence of a complex model that contains as much linguistic information as required by the final application and, additionally, in different languages.

*Localization*: the capability for providing a subset of linguistic descriptions to account for the linguistic realization of an ontology in different natural languages and representing term variants within one language and cultural specificities among different languages.

*Interoperability*: the flexibility of interoperating with existing standards for the representation of lexical and terminological information. By interoperating with standard models, there also exists the possibility for the model of interchanging knowledge with the standards and being extended with further linguistic description elements, if so required by the final application.

*Accessibility*: the fact of being implemented in a syntax or representation language that can provide tool support available to manage it, as well as access to external resources from which information can be obtained to semi-automatically support the model.

## *4.3.1 LIR Overview*

This section presents the *Linguistic Information Repository* or LIR, a model that has been created with the twofold purpose of fulfilling the needs of portability and association of multilingual information to domain ontologies, on the one hand, and adapting ontologies to the needs of the languages involved in the localization activity, on the other.

The LIR has been implemented as an ontology in OWL. Its main purpose is not to provide a model for a lexicon of a language but to cover a subset of linguistic description elements that account for the linguistic realization of a domain ontology in different natural languages. A complete description of the current version of the LIR can be found in (Montiel-Ponsoda et al. 2008; Montiel-Ponsoda 2011).

The lexical and terminological information captured in the LIR is organized around the `LexicalEntry` class. Lexical entry is considered a union of word form (`Lexicalization`) and meaning (`Sense`). This ground structure has been inspired by the Lexical Markup Framework (LMF). The compliance with this standard is important for two main reasons: (a) Links to lexicons modeled according to this standard can be established, and (b) the LIR can be flexibly

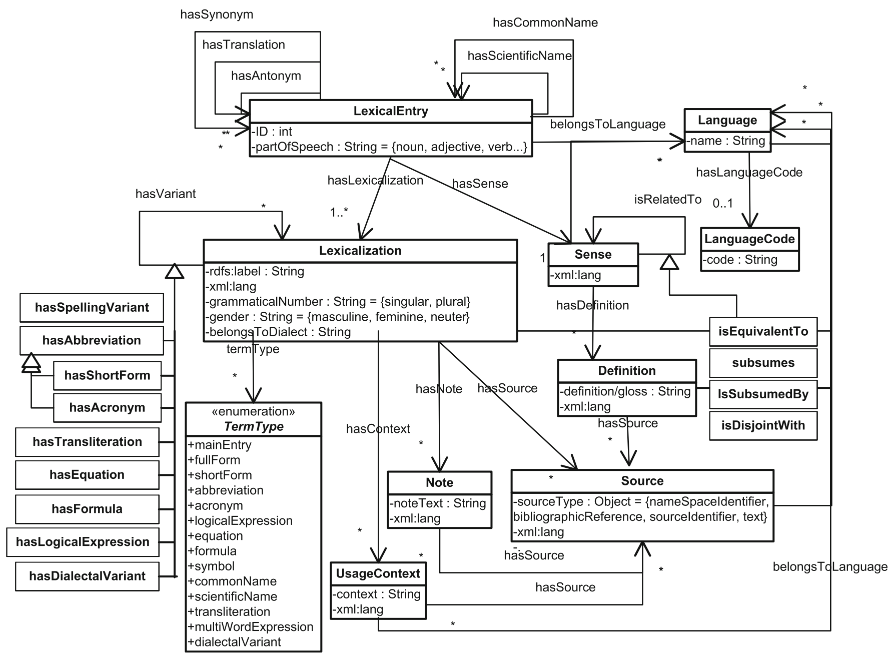

**Fig. 4.2** Diagram of LIR ancillary classes

extended with modular extensions of the LMF (or standard-compliant) modeling specific linguistic aspects, such as deep morphology or syntax, not dealt by LIR in its present stage. For more details on the interoperability of the LIR with further standards see (Peters et al. 2010).

The rest of the classes that make up the LIR are `Language`, `Definition`, `Source`, `Note`, and `UsageContext` (see Fig. 4.2). These can be linked to the `Lexicalization` and `Sense` classes. Each lexicalization is associated to one sense. The sense class represents the meaning of the ontology concept in a given language. It has been modeled as an empty class because its purpose is to point to other resources in which that sense is captured. The meaning of the concept in a certain language (which may not completely overlap with the formal description of the concept in the ontology) is "materialized" in the definition class, i.e., is expressed in natural language. The `UsageContext` gives us information about how a word behaves syntactically in a certain language by means of examples. Source information can be attached to any class in the model (`Lexicalization`, `Definition`, etc.), and, finally, the `Note` class has been meant to include any information about language specificities, connotations, style, register, etc., and can be related to any class. By determining the `Language` of a lexical entry, we can ask the system to display only the linguistic information associated to the ontology belonging to a given language.

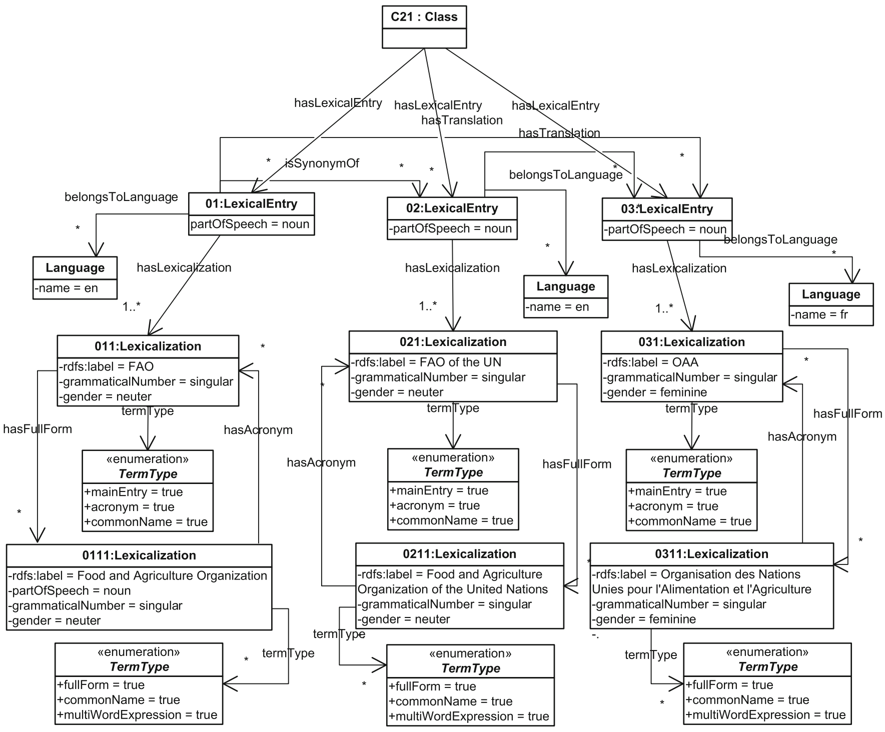

**Fig. 4.3** LIR example usage within a single language

## 4.3.2 *Uses and Benefits*

The main benefit of the LIR model is that it provides a very granular specification of relationships between elements of an ontology. In particular, it identifies well-defined relationships among the linguistic descriptions used to represent ontological concepts, specifically:

- Well-defined relations within lexicalizations in one language
- Well-defined relations within lexicalizations across languages

Both cases are illustrated in the following. The example in Fig. 4.3 concerns the establishment of relations among term variants belonging to the same language. Specifically, this case exemplifies the use of various acronyms and full forms attached to one and the same concept. Three lexical entries (01:LexicalEntry, 02: LexicalEntry, and 03:LexicalEntry) are associated with the same concept (C21: Class), which means that they are terms that identify one and the same concept. Two lexical entries (01:LexicalEntry and 02:LexicalEntry) belong to English, whereas the third lexical entry (03:LexicalEntry) belongs to French. The two English lexical entries are considered synonyms, and both are translations of the

French lexical entry. Each lexical entry contains two lexicalizations. For example, 01:LexicalEntry includes 011:Lexicalization and 0111:Lexicalization, whose labels are FAO and Food and Agriculture Organization, respectively. FAO is the acronym for Food and Agriculture Organization, and, moreover, it is considered the main entry. FAO of the UN and Food and Agriculture Organization of the United Nations are deemed synonyms of FAO and Food and Agriculture Organization. Both lexical entries (01:LexicalEntry and 02:LexicalEntry) are translations of OAA and Organisation des Nations Unies pour l'Alimentation et l'Agriculture in the French language. Thanks to LIR it is possible to retrieve synonyms within the same language associated with the same concept and distinguish different term types such as acronyms and full forms.

The second example highlights the possibility given by the LIR model to represent scientific names and use them across languages (scientific names are in Latin and are internationally accepted over scientific communities). Variants in the same language (e.g., Buffaloes (syncerus)) can therefore be connected to the same scientific term, such as the English and Japanese translations. We have illustrated in Fig. 4.4 how the concept buffaloes (C133:Class) has four lexical entries associated (01:LexicalEntry, 02:LexicalEntry, 03:LexicalEntry, and 04:LexicalEntry). Two of them belong to the English language and contain synonymous lexicalizations (011: Lexicalization and 021:Lexicalization).

Then, we have a lexicalization in Latin that represents the scientific name, and it is accordingly related with the rest of lexical entries by means of the object property `hasScientificName`. Finally, 04:LexicalEntry belongs to the Japanese language, which is also the common denomination in Japanese of the *Syncerus caffer* scientific name and, at the same time, the translation of the two lexicalizations in English.

To conclude, we refer to the *LabelTranslator* NeOn plugin, a translation-supporting tool (Espinoza et al. 2008) that provides semi-automatically translations

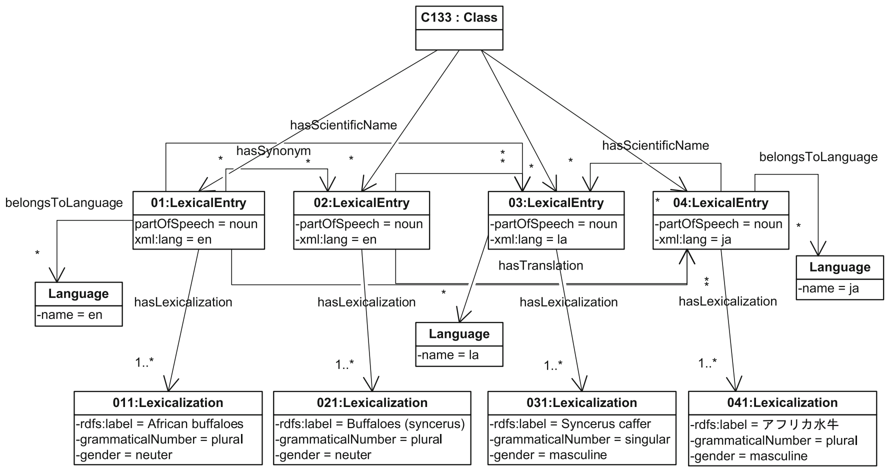

**Fig. 4.4** LIR example of cross-language usage

for ontology lexicalizations. Currently, the languages supported by the plugin are Spanish, English, and German. Once translations are obtained for the labels of the original ontology, they are stored in the LIR. However, if the system does not support the language combination in which we are interested, we can still use this system to take advantage of the LIR application programming interface or API implemented in the NeOn Toolkit. In this sense, the needed linguistic information can be introduced manually.

## 4.4 Collaborative Ontology Design Ontology (C-ODO) Light

Authoring and maintaining Semantic Web ontologies is generally not an individual, monolithic activity but is intrinsically grounded on social and collaborative processes, more so when ontologies are configured in a networked architecture. Continuous interaction between knowledge engineers and domain experts is key and so is that with resource providers whenever reuse or re-engineering enters the life cycle.

However, without dedicated tool support, collaboration may occur across general-purpose software tools and communication channels, in which case a manual effort is required to coordinate and bring the outcome of these activities together. Thus, on one hand, tool support to ontology engineering activities (e.g., *reusing* existing ontologies and design patterns; *re-engineering* thesauri, lexica, and database schemas; *validating* the outcome) is required. On the other hand, tools are often unable to support these activities in a *collaborative* setting, e.g., aiding the discussion and consensus-based assessment of an ontology element and the rationale behind it. Among other reasons, this can also be ascribed to an inadequate requirement analysis describing the actual processes and data involved therein, and the lack of a conceptual framework that formally expresses these notions so that they can be unified and reasoned upon.

### *4.4.1 C-ODO Light Overview*

C-ODO Light (aka *codolight*) is one such formal knowledge framework. It is a pattern-based OWL-DL ontology network that provides a metamodel for describing collaborative ontology projects (Gangemi et al. 2007). C-ODO Light was designed so as to take into account requirements deriving from the experience with formal models for describing ontology projects and tools, as well as existing controlled vocabularies.

In particular, the network displays the following features:

1. The ability to formalize ontology design tool descriptions in terms of input/output data (knowledge types), functionalities, interface objects, and interaction patterns

2. Smooth integration between human-oriented and tool-oriented descriptions of ontology design aspects
3. Alignment to existing vocabularies such as DOAP, OMV, etc.
4. Light axiomatization, e.g., no use of anonymous classes in restrictions
5. Modular development by pattern-based design (cf. Chap. 3), in compliance with the `ontologydesignpatterns.org` practices

Additionally, the *codolight* core is extended to support specific ontology application tasks, such as:

1. Browsing semantic data about ontology projects, tools, data, repositories, solutions, discussion, evaluation, etc.
2. Searching and selecting design components based on design aspects, knowledge types, individual needs, user profiles, etc.
3. Creating design configuration interfaces that aid or automate task 2
4. Help collecting ontology requirements, design functionalities, and ontology application tasks for an ontology project
5. Providing a shared network of vocabularies to create/query/reason on annotations and data related to ontology projects, including integration between annotations of heterogeneous provenance, such as those coming from collaborative discussions and change

It is here anticipated that part of these tasks are implemented within NeOn in the form of the *Kali-ma* tool, to be described in Chap. 15.

### *4.4.2 Structure*

The C-ODO Light network of ontologies is organized as a layered architecture, where these layers are connected with different types of bindings. Besides the two bottom layers that define the main structure, there are three additional layers that bridge it with existing applications, vocabularies, and functionalities:

*Pattern layer*: It contains reusable content ontology design patterns (Content ODP) (Presutti and Gangemi 2008) that include, e.g. *sequence*, *partof*, *situation*, *collectionentity,* and so on. The patterns are reused in the design of the ontologies constituting the core architecture of *codolight*.

*Core codolight layer*: It contains the nine modules of the *codolight* core network of ontologies, centered around the *codkernel* module in the center, along with modules *coddata*, *codprojects*, *codworkflows*, *codarg*, *codsolutions*, *codtools*, *codinterfaces*, and *codinteraction* importing *codkernel*.

*Plugin layer*: It consists of the modules containing the descriptions of the NeOn Toolkit plugins related to ontology design, formalized in OWL by reusing the codolight vocabulary and some of the alignment modules.

*Categorization layer*: It consists of the modules containing the definition of the design aspects according to which tools, knowledge types, and functionalities are organized, as well as the closure of inferences derived by the application of reasoners to the previous layers.

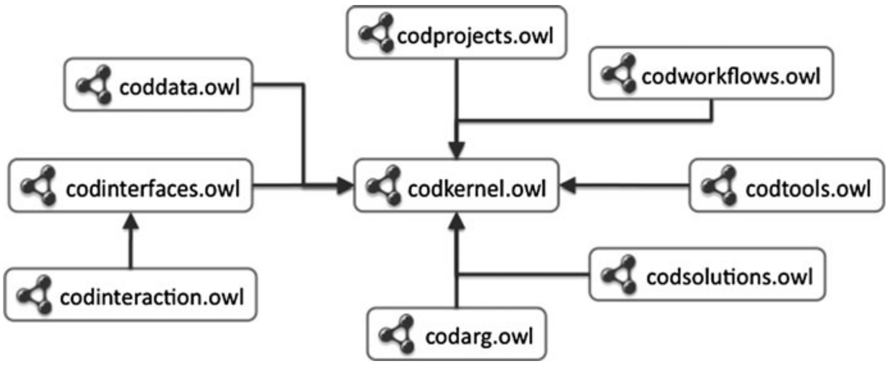

**Fig. 4.5** Core C-ODO Light corolla architecture

*Alignment layer*: It consists of the modules containing mapping axioms between *codolight* and related vocabularies, currently: OMV, DOAP, FOAF, NeOn Access Rights model, NeOn OWL metamodel, NeOn OWL2 metamodel, and the Software Ontology Model.

Each layer is in its turn an ontology network with its own architecture. In particular, the core *codolight* layer network encodes the main aspects of ontology design by following an architectural ontology design pattern called *corolla*. The floral metaphor for the corolla pattern, shown in Fig. 4.5, suggests an overall shape for the network composed of a *kernel* module, which includes the definition of core concepts of the domain of interest, and a set of *petal* modules, each defining a specific aspect of the same domain.

The corolla pattern minimizes dependencies and enforces loose coupling between the modules of an ontology network: In the *codolight* core example, all modules, but *codinteraction* (described below), directly depend on the kernel module exclusively. Also, the modules are built so that their structure suggests an organization of the network, by which different aspects of the domain of interest are represented by each petal module. The criterion by which an ontology network can be broken apart into a corolla can be: *What are the main aspects of the domain described by the ontology network?*

The kernel module defines core concepts, shared by all aspects. As shown in Fig. 4.6, axiomatization is minimal at the kernel level, i.e., although the basic classes of collaborative ontology design are defined, only a minimum set of new properties, or restrictions holding between them, is asserted. It is up to petal modules to refine the axiomatization of at least one of the core concepts each, thus adding details for at least one aspect.

All classes defined in the kernel module are specializations of classes from the pattern layer (such as `DesignFunctionality` subsuming `Task` from the `taskrole` pattern[15] or `KnowledgeResource`, `FormalExpression`,

[15] http://www.ontologydesignpatterns.org/cp/owl/taskrole.owl

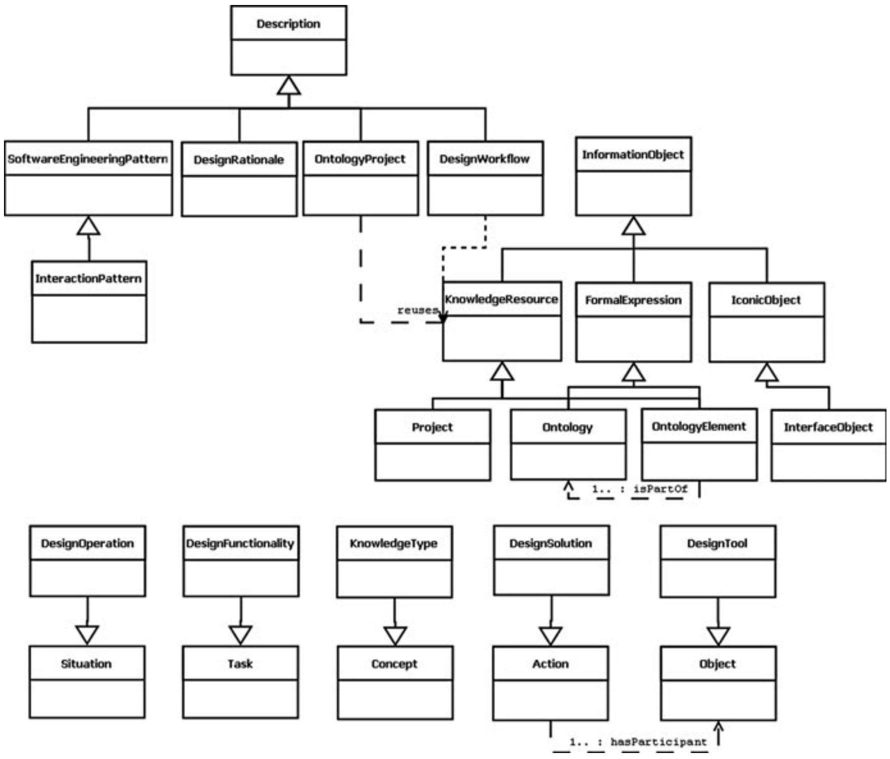

**Fig. 4.6** C-ODO Light kernel class diagram

`IconicObject` all subsuming `InformationObject` from the `intensionextension` pattern[16]), thus inheriting the structure defined by equality over shared classes.

The following petal modules are defined for *codolight*:

*Data* (*coddata*)*:* contains the main notions that classify the data managed when designing an ontology: ontologies, ontology elements, Knowledge Organization Systems (KOS), KOS elements, rules, modules, encoding syntaxes, and more. For each class of knowledge resources, a `knowledge-type` instance is provided.

*Projects* (*codprojects*): contains the minimal vocabulary for representing ontology design projects and their executions. An ontology project is here taken as a social entity, whose computational counterpart (e.g., a project created in the NeOn Toolkit) is a software entity that collects resources and descriptions related to an ontology project.

*Workflows* (*codworkflows*): contains classes and properties to represent workflows from within ontology projects: collaborative workflows, accountable agents, need for an agent or a design functionality, etc.

[16] http://www.ontologydesignpatterns.org/cp/owl/intensionextension.owl

*Argumentation* (*codarg*): contains the basic classes and properties to represent argumentation concepts: arguments, threads, ideas, positions, rationales, etc.

*Solutions* (*codsolutions*): contains classes and properties to represent ontology design solutions: competency questions, ontology design patterns, ontology requirements, unit tests, etc.

*Tools* (*codtools*): contains classes and properties to represent ontology design tools: tools, pieces of code, code entities, computational tasks, input and output data relations, etc.

*Interfaces* (*codinterfaces*): contains classes and properties that represent some typical user interface entities, such as interface objects, panes, and windows.

*Interaction* (*codinteraction*): contains classes and properties that represent some typical entities related to human-computer and human-ontology interaction, e.g., user types, computational tasks, and workflows. These can in turn be combined so as to construct interaction pattern models.

One advantage of employing this aspect-oriented architecture is *selective extensibility*. A software application intended to exploit only a given subsystem of ontology life cycle management, such as reasoning on the usage of interaction patterns and user interface widgets, can import only the *codinterfaces* and *codinteraction* modules. Possibly, they can also extend these modules with ontologies that model further additional interaction patterns or GUI elements originally not intended for the application domain at hand.

### 4.4.3 Alignments

This section provides an insight on some alignments that hold between *codolight* and other vocabularies that are widely used on the Semantic Web or are introduced as part of the methodology described by this book (c.f. Chap. 2), such as OMV that is described in this very chapter.

*OWL*[17]. The alignments between *codolight* and the OWL language constructs consider three different vocabularies:

- The original RDF, RDFS, and OWL vocabularies from the W3C
- The OWL1 metamodel designed for NeOn
- The OWL2 metamodel also designed for NeOn

The reason why so many different vocabularies represent entities from a same language is mainly due to the pragmatic evolution of semantic technologies. The original vocabularies by W3C are not extremely detailed in distinguishing the constructs available in OWL (and RDF, RDFS); e.g., it is difficult to describe existential restrictions explicitly, because these are just instances of `owl:Restriction`. On the other hand, W3C vocabularies are implemented in all APIs and tools for ontology engineering, so in order to maximize interoperability,

[17] http://www.ontologydesignpatterns.org/cpont/codo/owl22codo.owl

an ontology design vocabulary like *codolight* must be aligned to the main data vocabularies. The OWL metamodels developed in NeOn try to overcome the referential coarseness of OWL constructs, e.g., by providing the class `owlodm1:ExistentialRestriction`. On the other hand, these metamodels are not intended to be a replacement for the W3C OWL data model.

*Ontology Metadata Vocabulary (OMV)*[18]. The OMV described in this chapter is a vocabulary for annotating ontologies with time, authors, tools, languages, etc., and it is used to provide support for ontology registries. However, from a design viewpoint, the semantics of OMV metadata are potentially compatible with those of other metamodels, so this alignment aids metadata interoperability. Only subsumption alignments occur, i.e., OMV classes and properties are subclasses or subproperties of those from the *codolight* pattern and core layers. Aligned OMV classes of interest include `omv:OntologySyntax`, `omv:FormalityLevel`, `omv:OntologyEngineeringTool`, and `omv:OntologyTask`. Aligned OMV properties of interest include `omv:hasContributor`, `omv:hasCreator`, `omv:useImports`, and `omv:isIncompatibleWith`.

*Description of a Project (DOAP)*[19]. DOAP is a vocabulary for creating profiles of software projects[20] with time, authors, FOAF (Friend of a Friend) vocabulary profiles, etc. The `doap:Project` notion addressed here is computational and not social; therefore, it has been aligned as equivalent to `codkernel:Project`. Subsumption mappings occur between `doap:Repository` and `collectionentity:Collection`, and between `doap:Version` and `coddata:Annotation`. Direct mappings occur between FOAF and the *codolight* pattern layer, either by equivalence (`foaf:Agent` and `agentrole:Agent`) or by subsumption (`foaf:topic` and `topic:hasTopic`; `foaf:Document` and `intensionextension:InformationObject`; `foaf:member` and `collectionentity:hasMember`).

*NeOn Access Rights Model*[21]. This ontology representing access control policies and related entities uses three different vocabularies, i.e., *accessRights*[22], *ar-entities*[23], and *ar-agents*[24]. Entities from these vocabularies, such as `Action`, `Agent`, and `Content`, map to *codolight* via subsumption alignments. The `Right` class, being a conceptualization of its holder in the access policies context, subsumes the `Description` class from the `description` design pattern.

*Software Ontology Model (SOM)*[25]. The SOM is designed to represent entities in the object-oriented programming model (Tappolet et al. 2010). Given the class

[18] http://www.ontologydesignpatterns.org/cpont/codo/omv2codo.owl

[19] http://www.ontologydesignpatterns.org/cpont/codo/doap2codo.owl

[20] http://trac.usefulinc.com/doap

[21] http://www.ontologydesignpatterns.org/cpont/codo/accessrights2codo.owl

[22] http://www.uni-koblenz.de/~bercovici/owl/2008/7/accessRight.owl

[23] http://www.uni-koblenz.de/~bercovici/owl/2008/7/entity.owl

[24] http://www.uni-koblenz.de/~schwagereit/owl/agents.owl

[25] http://www.ontologydesignpatterns.org/cpont/codo/som2codo.owl

subtree for the `som:Entity` class, which includes functions, methods, classes, and packages, the parent class aligns to `codtools:CodeEntity` by equivalence.

## 4.5 Conclusions

A methodology for managing ontology networks is best designed if formal models of the resources and processes involved come along with it. To that end, the NeOn Methodology proposes three stand-alone models, i.e., the OMV, LIR, and C-ODO Light ontologies, that can nonetheless be interconnected in order to represent and reason on the structural, linguistic, and engineering aspects of ontology life cycle. Ontology alignments are provided across these models to ensure logic interoperability, without hampering their stand-alone usage possibilities. These ontologies also serve as a back end for software applications provided as plugins for the NeOn Toolkit (see Chaps. 13, 14, 15), which treat each ontology separately to serve dedicated phases and processes in ontology management.

## References

Arpírez J, Gómez-Pérez A, Lozano-Tello A, Pinto HS (2000) Reference ontology and (ONTO) 2 agent: the ontology yellow pages. Knowl Inf Syst 2:387–412

Buitelaar P, Declerck T, Frank A, Racioppa S, Kiesel M, Sintek M, Engel R, Romanelli M, Sonntag D, Loos B, Micelli V, Porzel R, Cimiano P (2006) Linginfo: design and applications of a model for the integration of linguistic information in ontologies. In: Proceedings of the OntoLex 2006 workshop: interfacing ontologies and lexical resources for semantic web technologies, Genoa

Buitelaar P, Cimiano P, Haase P, Sintek M (2009) Towards linguistically grounded ontologies. In: Proceedings of the 6th annual European semantic web conference (ESWC2009), Heraklion, pp 111–125

Cimiano P, Haase P, Herold M, Mantel M, Buitelaar P (2007) LexOnto: a model for ontology lexicons for ontology-based nlp. In: Proceedings of the OntoLex07 workshop at the ISWC07, Busan

d'Aquin M, Haase P, Rudolph S, Euzenat J, Zimmermann A, Dzbor M, Iglesias M, Jacques Y, Caracciolo C, Buil-Aranda C, Gómez-Pérez J (2008) NeOn formalisms for modularization: syntax, semantics, algebra. Technical report D1.1.3, Open University

Espinoza M, Gómez-Pérez A, Mena E (2008) Enriching an ontology with multilingual information. In: Proceedings of the 5th annual of the European semantic web conference (ESWC 2008), Tenerife, pp 333–347

Fridman Noy N, Guha RV, Musen MA (2005) User ratings of ontologies: who will rate the raters? In: Proceedings of the AAAI 2005 spring symposium on knowledge collection from volunteer contributors, Stanford, CA, USA

Fridman Noy N, Griffith N, Musen MA (2008) Collecting community-based mappings in an ontology repository. In: Proceedings of 7th international semantic web conference´08. Springer, Karlsruhe

Gangemi A, Pisanelli DM, Steve G (1999) An overview of the ONIONS project: applying ontologies to the integration of medical terminologies. Data Knowl Eng 31(2):183–220

Gangemi A, Lehmann J, Presutti V, Nissim M, Catenacci C (2007) C-ODO: an OWL meta-model for collaborative ontology design. In: Fridman Noy N, Alani H, Stumme G, Mika P, Sure Y, Vrandecic D (eds) CKC, CEUR-WS.org

Gardiner T, Horrocks I, Tsarkov D (2006) Automated Benchmarking of Description Logic Reasoners. In Parsia B, Sattler U, Toman D (eds) Proc. of the Int. Workshop on Description Logics (DL'06), Windermere Lake District, UK. Volume 189 of CEUR., Lake District, UK 167–174

ISO 16642 (2003) Terminological markup framework in computer applications in terminology. Technical report, International Organization for Standardization (ISO). URL http://www.loria.fr/projets/TMF/

ISO 24613 (2006) Lexical markup framework in language resource management. Technical report, International Organization for Standardization (ISO). URL http://lirics.loria.fr/doc_pub/LMF%20rev9%2015March2006.pdf

Jarrar M (2005) Towards methodological principles for ontology engineering. PhD thesis, Vrije Universiteit Brussel, Brussels

Lozano-Tello A, Gómez-Pérez A (2004) ONTOMETRIC: a method to choose the appropriate ontology. J Database Manag 15(2)

Miles A, Matthews B, Beckett D, Brickley D, Wilson M, Rogers N (2005) SKOS: a language to describe simple knowledge structures for the web. In: Proceedings of the XTech conference 2005, Amsterdam

Montiel-Ponsoda E (2011) Multilingualism in ontologies: multilingual lexico-syntactic patterns for ontology modeling and linguistic information repository for ontology localization. PhD thesis, Universidad Politécnica de Madrid, Madrid

Montiel-Ponsoda E, Aguado de Cea G, Suárez-Figueroa MC, Palma R, Peters W, Gómez-Pérez A (2007) LexOMV: an OMV extension to capture multilinguality. In: 6th international semantic web conference. In Workshop Ontolex07, Busan

Montiel-Ponsoda E, Peters W, Aguado de Cea G, Espinoza M, Gómez-Pérez A, Sini M (2008) Multilingual and localization support for ontologies. Technical report, D2.4.2 NeOn project deliverable

Montiel-Ponsoda E, Aguado de Cea G, Gómez-Pérez A, Peters W (2010) Enriching ontologies with multilingual information. J Nat Lang Eng 17(3):283–309

NISO (2004) Understanding metadata. NISO Press, National Information Standards Organization. Available at http://www.niso.org/publications/press/UnderstandingMetadata.pdf

Palma R (2009) Ontology metadata management in distributed environments. PhD thesis, Universidad Politécnica de Madrid

Palma R, Haase P (2005) Oyster – sharing and re-using ontologies in a peer-to-peer community. In: International semantic web conference, Galway, pp 1059–1062

Palma R, Hartmann J, Haase P (2008) OMV – ontology metadata vocabulary for the semantic web. Technical report, Universidad Politécnica de Madrid, University of Karlsruhe. Version 2.4. Available at http://omv.ontoware.org/

Paslaru Bontas E, Mochol M, Tolksdorf R (2005) Case studies on ontology reuse. In: Proceedings of the IKNOW05 international conference on knowledge management, Graz

Peters W, Gangemi A, Villazón-Terrazas B (2010) Modelling and re-engineering linguistic/terminological resources. Technical report, D2.4.4 NeOn project deliverable

Pinto HS, Martins JP (2001) A methodology for ontology integration. In: Proceedings of the international conference on knowledge capture K-CAP01, Victoria

Presutti V, Gangemi A (2008) Content ontology design patterns as practical building blocks for web ontologies. In: ER '08: proceedings of the 27th international conference on conceptual modeling. Springer, Berlin/Heidelberg, pp 128–141

Russ T, Valente A, Macgregor R (1999) Practical experiences in trading off ontology usability and reusability. In: Proceedings of the 12th workshop on knowledge acquisition, modeling and management (EKAW'99), Banff, pp 16–21

Tappolet J, Kiefer C, Bernstein A (2010) Semantic web enabled software analysis. J Web Semant 8(2–3):225–240

Ungrangsi R, Simperl E (2008) OMEGA: an automatic ontology metadata generation algorithm. In: 16th international conference on knowledge engineering, knowledge management and knowledge patterns. Springer, Berlin/Heidelberg/New York

Uschold M, Healy M, Williamson K, Clark P, Woods S (1998) Ontology reuse and application. In: Proceedings of the international conference on formal ontology and information systems FOIS98, Trento

Vossen P (1998) Introduction to EuroWordNet. In Ide N, Greenstein D, Vossen P (eds) Special issue on EuroWordNet, vol 32(2–3), pp 73–89

Wang Y, Haase P, Palma R (2007) D1.4.1: Prototypes for managing networked ontologies. Technical report D1.4.1, University of Karlsruhe; NeOn deliverable. URL http://www.neon-project.org/