1. Session: Ontological Engineering State of the Art

1.1. Paper: Essentials In Ontology Engineering: Methodologies, Languages, And Tools

Mari Carmen Suárez-Figueroa, PhD, mcsuarez@fi.upm.es
Raúl García-Castro, PhD, rgarcia@fi.upm.es
Boris Villazón Terrazas, PhD, bvillazon@fi.upm.es
Asunción Gómez-Pérez, PhD, asun@fi.upm.es
Ontology Engineering Group, Universidad Politécnica Madrid

Abstract

In the beginning of the 90s, ontology development was similar to an art: ontology developers did not have clear guidelines on how to build ontologies but only some design criteria to be followed. Work on principles, methods and methodologies, together with supporting technologies and languages, made ontology development become an engineering discipline, the so-called Ontology Engineering. Ontology Engineering refers to the set of activities that concern the ontology development process and the ontology life cycle, the methods and methodologies for building ontologies, and the tool suites and languages that support them. Thanks to the work done in the Ontology Engineering field, the development of ontologies within and between teams has increased and improved, as well as the possibility of reusing ontologies in other developments and in final applications. Currently, ontologies are widely used in (a) Knowledge Engineering, Artificial Intelligence and Computer Science, (b) applications related to knowledge management, natural language processing, e-commerce, intelligent information integration, information retrieval, database design and integration, bio-informatics, education, and (c) the Semantic Web, the Semantic Grid, and the Linked Data initiative. In this paper, we provide an overview of Ontology Engineering, mentioning the most outstanding and used methodologies, languages, and tools for building ontologies. In addition, we include some words on how all these elements can be used in the Linked Data initiative.

Keywords: Ontology engineering, ontology development methodologies, ontology languages, ontology tools.

1.1.1. Introduction

Ontologies play an important role for many knowledge-intensive applications, since they provide formal models of domain knowledge that can be exploited in different ways. Currently, ontologies are

used in (a) Knowledge Engineering, Artificial Intelligence and Computer Science, (b) applications related to knowledge management, natural language processing, e-commerce, intelligent information integration, information retrieval, database design and integration, bio-informatics, education, and (c) the Semantic Web, the Semantic Grid, and the Linked Data initiative.

During the last two decades, increasing attention has been focused on ontologies and their development. Indeed, ontology development has become an engineering discipline, Ontology Engineering, which refers to the set of activities that concern the ontology development process and the ontology life cycle, the methods and methodologies for building ontologies, and the tool suites and languages that support them.

When ontologies are going to be built, several basic questions arise related to the methodologies, languages, and tools to be used in their development processes:

- Which methods and methodologies can be used for building ontologies? Which activities are performed when building ontologies with a particular methodology? Does any methodology support building ontologies cooperatively? Which is the life cycle of an ontology that is developed with a specific methodology?
- Which language(s) should be used to implement an ontology? What expressiveness has an ontology language? What are the inference mechanisms attached to an ontology language? Is the language chosen appropriate for exchanging information between different applications? Does the language ease the integration of the ontology in an application?
- Which tool(s) give/s support to the ontology development process? Does the tool have an inference engine? How can applications interoperate with ontology servers and/or use the ontologies that we have developed?

Along this paper, we present the basics about ontologies, and show what methodologies, languages, and tools are available to give support to different activities of the ontology development process. The content of this paper can help practitioners and researchers in this field to obtain answers to the some of the previous questions. In addition, we include some words on how these elements (methodologies, languages, and tools) can be used in the Linked Data initiative. First, in Section 2, we define the world 'ontology' and briefly describe its main components. After that, in Section 3, we enumerate methodologies commonly used for building ontologies. In Section 4, we summarize the most used ontology languages and in Section 5, we present some tools used in the ontology development process. In Section 6, we provide some guidelines on how to publish linked data. Finally, we present some conclusions.

1.1.2. Ontology definition and its main components

The word ontology was taken from Philosophy, where it means a systematic explanation of being. There are many definitions about what an ontology is and such definitions have changed and evolved over the years. However, Studer and colleagues (Studer et al., 1998) provide one of the most well known definitions: "*An ontology is a formal, explicit specification of a shared conceptualization. Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used, and the*

constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine-readable. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group".

Ontologies can be modelled with different knowledge modelling techniques and they can be implemented in various kinds of languages based on different knowledge representation formalisms. It is important to mention here that there are connections and implications between the knowledge modelling components (concepts, roles, etc.) used to build an ontology, the knowledge representation paradigms (frames, description logics, logic) used to represent formally such components, and the languages used to implement the ontologies under a given knowledge representation paradigm. However, they share the following minimal set of components:

- **Classes** represent concepts, which are taken in a broad sense. For instance, in the domain of Energy Efficiency at Buildings, concepts are *Building*, *Door*, *Window*, *Device*, *Sensor*, etc. Classes in the ontology are usually organized in taxonomies through which inheritance mechanisms can be applied. We can represent a taxonomy of sensors (*Scanning Sensor*, *Optical Sensor*, *Touch Trigger Sensor*, etc.) or different types of doors in buildings (*Inner Door*, *Outer Door*, *Sliding Door*, *Rotating Door*, or *Strongroom Door*).
- **Relations** represent a type of association between concepts of the domain. They are formally defined as any subset of a product of n sets, that is: $R \subset C1 \times C2 \times ... \times Cn$. Ontologies usually contain binary relations. The first argument is known as the domain of the relation, and the second argument is the range. For instance, the binary relation *locatedIn* has the concept *Building* as its domain and the concept Location as its range; in addition, this relation can have the concept *Device* as domain. Binary relations are sometimes used to express concept attributes (aka slots). Attributes are usually distinguished from relations because their range is a datatype, such as string, number, etc., while the range of relations is a concept.
- **Formal axioms**, according to (Gruber, 1993), serve to model sentences that are always true. They are normally used to represent knowledge that cannot be formally defined by the other components. In addition, formal axioms are used to verify the consistency of the ontology itself or the consistency of the knowledge stored in a knowledgebase. Formal axioms are very useful to infer new knowledge. An axiom in the Energy Efficiency at Buildings domain could be that it is not possible to build a public building without a fire door (based on legal issues).
- **Instances** are used to represent elements or individuals in an ontology.

1.1.3. Foremost methodologies for building ontologies

METHONTOLOGY, On-To-Knowledge, and DILIGENT were up to 2009 the most referred methodologies for building ontologies. These methodologies mainly include guidelines for single ontology construction ranging from ontology specification to ontology implementation and they are mainly targeted to ontology researchers. In contrast to the aforementioned approaches, a new methodology, called the NeOn Methodology, suggests pathways and activities for a variety of scenarios, instead of prescribing a rigid workflow.

In this section, we summarize the four abovementioned methodologies, describing in more detailed the current trend in ontology building presented in the NeOn Methodology.

- **The NeOn Methodology** (Suárez-Figueroa, 2010) for building ontology networks is a scenario-based methodology that supports a knowledge reuse approach, as well as collaborative aspects of ontology development and dynamic evolution of ontology networks in distributed environments. The key assets of the NeOn Methodology are:
    - A set of nine scenarios for building ontologies and ontology networks, emphasizing the reuse of ontological and non-ontological resources, the reengineering and merging, and taking into account collaboration and dynamism.
    - The NeOn Glossary of Processes and Activities, which identifies and defines the processes and activities carried out when ontology networks are collaboratively built by teams.
    - Methodological guidelines for different processes and activities of the ontology network development process, such as the reuse and reengineering of ontological and non-ontological resources, the ontology requirements specification, the ontology localization, the scheduling, etc. All processes and activities are described with (a) a filling card, (b) a workflow, and (c) examples.

The set of nine scenarios for building ontologies and ontology networks can be summarized as follows:

- *Scenario 1: From specification to implementation*. The ontology network is developed from scratch (without reusing existing resources). Developers should specify ontology requirements (Suárez-Figueroa et al., 2009). After that, it is advisory to carry out a search for potential resources to be reused. Then, the scheduling activity (Suárez-Figueroa et al., 2010) must be performed, and developers should follow the plan to develop the ontology network.
- *Scenario 2: Reusing and re-engineering non-ontological resources (NORs)*. Developers should carry out the NOR reuse process for deciding, according to the ontology requirements, which NORs can be reused to build the ontology network. Then, the selected NORs should be re-engineered into ontologies (Villazón-Terrazas et al., 2010).
- *Scenario 3: Reusing ontological resources*. Developers use ontological resources (ontologies as a whole, ontology modules, and/or ontology statements) to build ontology networks.
- *Scenario 4: Reusing and re-engineering ontological resources*. Ontology developers reuse and re-engineer ontological resources.
- *Scenario 5: Reusing and merging ontological resources*. This scenario arises when several ontological resources in the same domain are selected for reuse, and developers wish to create a new ontological resource with the selected resources.
- *Scenario 6: Reusing, merging and re-engineering ontological resources*. Ontology developers reuse, merge, and re-engineer ontological resources. This scenario is similar to Scenario 5, but here developers decide to re-engineer the set of merged resources.
- *Scenario 7: Reusing ontology design patterns (ODPs)*. Ontology developers access repositories (e.g., http://ontologydesignpatterns.org/) to reuse ODPs.

- *Scenario 8: Restructuring ontological resources*. Ontology developers restructure (e.g., modularize, prune, extend, and/or specialize) ontological resources to be integrated in the ontology network.
- *Scenario 9: Localizing ontological resources*. Ontology developers adapt an ontology to other languages and culture communities, thus obtaining a multilingual ontology (Espinoza et al., 2009).

- **METHONTOLOGY** (Gómez-Pérez et al., 2003) enables the construction of ontologies at the knowledge level. It includes (a) the identification of the ontology development process (which tasks should be performed when building ontologies); (b) a life cycle based on evolving prototypes; and (c) some techniques to carry out management, development-oriented, and support activities. In addition, METHONTOLOGY includes a list of activities to be carried out during ontology reuse and re-engineering processes, but it does not provide detailed guidelines for such activities, nor does it consider different levels of granularity during the reuse of ontological resources (e.g., modules or statements). Moreover, METHONTOLOGY considers neither the reuse and re-engineering of non-ontological resources nor the reuse of ODPs.
- **The On-To-Knowledge methodology** (Staab et al., 2001) proposes to build ontologies taking into account how these are going to be used in knowledge management applications. The processes proposed by this methodology are the following: feasibility study, kickoff, where ontology requirements are identified, refinement, where a mature and application-oriented ontology is produced, evaluation, and maintenance. With respect to the reuse of knowledge resources, in the kickoff process it is mentioned that developers should look for potentially reusable ontologies. However, this methodology does not provide detailed guidelines for identifying such ontologies nor for reusing them. Besides, the methodology does not explicitly mention guidelines for the reuse and re-engineering of non-ontological resources, nor for the reuse of ontology design patterns.
- **The DILIGENT methodology** (Pinto et al., 2004) is intended to support domain experts in a distributed setting in order to engineer and evolve ontologies. This methodology is focused on collaborative and distributed ontology engineering. Its ontology development process includes the following five activities: building, local adaptation, analysis, revision, and local update. With regard to the reuse of knowledge resources, the methodology does not include guidelines for the reuse and re-engineering of existing knowledge resources.

1.1.4. Major ontology languages

Different ontology languages have different expressiveness and inference mechanisms, since the knowledge representation paradigms underlying all these languages are diverse. Therefore, one of the key decisions to take in the ontology development process is to select the language (or set of languages) in which the ontology will be implemented.

Next, we present an overview of the current specifications for ontology languages developed in the scope of the W3C Semantic Web Activity (http://www.w3.org/2001/sw/).

- **RDF**. RDF (Klyne and Carroll, 2004) stands for Resource Description Framework. It was developed by the W3C to create metadata for describing web resources and its data model is equivalent to the semantic networks formalism, consisting of three object types: resources, properties and statements.
- **RDF Schema**. The RDF data model does not have mechanisms for defining the relationships between properties and resources. This is the role of the RDF Vocabulary Description language (Brickley and Guha, 2004), also known as RDF Schema. RDF(S) is the term commonly used to refer to the combination of RDF and RDFS. Thus, RDF(S) combines semantic networks with frames but it does not provide all the primitives that are usually found in frame-based knowledge representation systems.
- **OWL**. OWL (Dean and Schreiber, 2004) is the result of the work of the W3C Web Ontology Working Group. This language derived from DAML+OIL (van Harmelen et al., 2001) and, as the previous languages, is intended for publishing and sharing ontologies in the Web. OWL is built upon RDF(S), has a layered structure and is divided into three sublanguages: OWL Lite, OWL DL and OWL Full. OWL is grounded on Description Logics (Baader et al., 2002) and its semantics is described in two different ways: as an extension of the RDF(S) model theory and as a direct model-theoretic semantics of OWL. Both of them have the same semantic consequences on OWL ontologies.
- **OWL 2**. OWL 2 (Motik et al., 2009) is an extension and revision of OWL that adds new functionality with respect to OWL; some of the new features are syntactic sugar (e.g., disjoint union of classes) while others offer new expressivity. OWL 2 includes three different profiles (i.e., sublanguages) that offer important advantages in particular application scenarios, each trading off different aspects of OWL's expressive power in return for different computational and/or implementational benefits. These profiles are:
    - OWL 2 EL that is particularly suitable for applications where very large ontologies are needed, and where expressive power can be traded for performance guarantees.
    - OWL 2 QL that is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to access the data directly via relational queries (e.g., SQL).
    - OWL 2 RL that is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to operate directly on data in the form of RDF triples.

OWL 2 provides two alternative ways of assigning meaning to OWL 2 ontologies: the Direct Semantics that assigns meaning directly to ontology structures and the RDF-Based Semantics that assigns meaning directly to RDF graphs.

- **SPARQL**. Even if it is not an ontology language, we mention SPARQL (Prud'hommeaux and Seaborne, 2008) here because it supports querying the previous languages. SPARQL allows performing queries over RDF data and, since both RDF-S and OWL are based in RDF, also over RDF-S and OWL ontologies. SPARQL can be used to express queries across diverse data sources and its syntax is similar to SQL to facilitate its adoption.

1.1.5. Leading ontology tools

The landscape of tools that manage and exploit ontologies is broad and covers from the creation of these ontologies to their storage or visualization.

Next, we describe the different dimensions in which semantic technologies can be classified according to their functionalities (see Figure 1); these dimensions are based in the Semantic Web Framework (García-Castro et al., 2008). Each dimension description contains the names of some relevant tools.

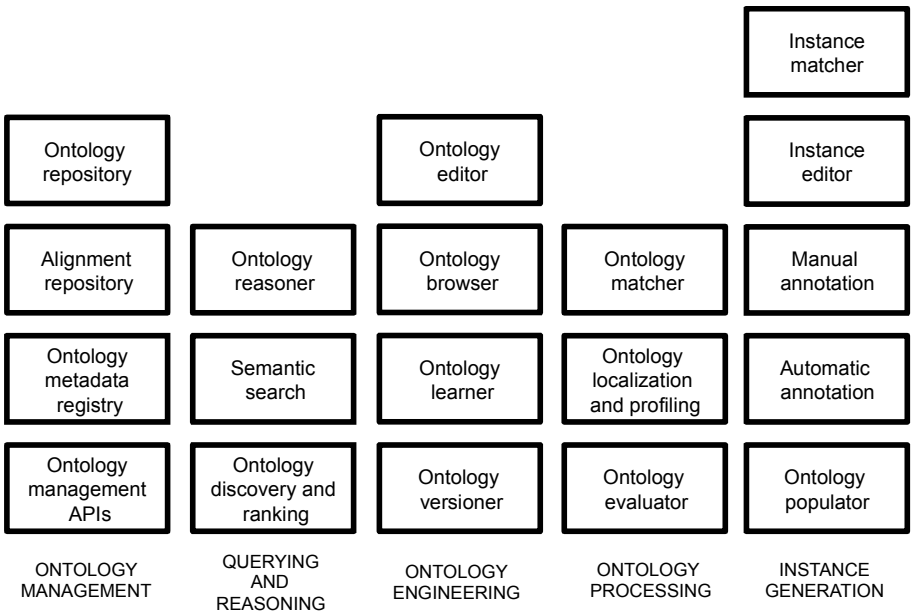

Figure 1: Semantic technology dimensions (adapted from García-Castro et al., 2008)

- **Ontology management**. This dimension includes components that manage ontology-related information.
  - The *Ontology repository* stores and accesses ontologies and ontology instances (e.g., 3Store, AllegroGraph, Corese, Hawk, Jena, Kowari, OWLIM, Sesame, Virtuoso Universal Server, 4store).
  - The *Alignment repository* stores and accesses alignments (e.g., Alignment Server, COMA++).
  - The *Ontology metadata registry* stores and accesses ontology metadata information (e.g., Oyster, SchemaWeb). This metadata information can be described using OMV (http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/downloads/75-omv), the Ontology Metadata Vocabulary.
  - The *Ontology management programming interfaces* provide programming interfaces for managing ontologies and ontology instances (e.g., OWL API, RDF2Go, SemWeb.NET, Pubby, Elda).
- **Querying and reasoning**. This dimension includes components that generate and process queries.

- The *Ontology reasoner* takes care of reasoning over ontologies and ontology instances (e.g., CEL, Cerebra Engine, FaCT ++, fuzzyDL, HermiT, KAON2, MSPASS, Pellet, QuOnto, RacerPro, SHER, SoftFacts, TrOWL).
- The *Semantic search* component takes care of the user interface for editing queries and of their corresponding processing (e.g., ARQ, Ginseng, K-Search, NLP-Reduce, Ontogator, PowerAqua, SemSearch).
- The *Ontology discovery and ranking* component finds appropriate views, versions or subsets of ontologies, and then ranks them according to some criterion (e.g., Swoogle, Watson, Sindice).

- **Ontology engineering**. This dimension includes components that provide functionalities to develop and manage ontologies.
  - The *Ontology editor* allows creating and modifying ontologies, ontology elements, and ontology documentation. These functionalities include a single element edition or a more advanced edition such as ontology pruning, extension or specialization (e.g., DODDLE, graphl, GrOWL, ICOM, IsaViz, NeOn Toolkit, Ontotrack, Powl, Protégé, SemanticWorks, SemTalk, SWOOP, TopBraid Composer).
  - The *Ontology browser* allows to visually browse an ontology (e.g., Brownsauce, BrowseRDF, Disco, /facet, Fenfire, Jambalaya, Longwell, mSpace, OINK, Ontosphere 3D, Ontoviz, OWLViz, RDF Gravity, Tabulator, TGVizTab, Welkin).
  - The *Ontology learner* acquires knowledge and generates ontologies of a given domain through some kind of (semi) automatic process (e.g., KEA, OntoGen, OntoLearn, Text2Onto, TERMINAE).
  - The *Ontology versioner* maintains, stores and manages different versions of an ontology (e.g., SemVersion).
- **Ontology processing**. This dimension includes components that process ontologies.
  - The *Ontology matcher* matches two ontologies and outputs some alignments. We can distinguish two types of such systems: those that generate alignments and those that use alignments for other tasks, such as merging or mediating (e.g., AgreeMaker, AMW, AROMA, ASMOV, AUTOMS, CMS, CODI, COMA, Ef2Match, Falcon-AO, Gerome, HMatch, Lily, MapOnto, Mapso, OLA, OntoBuilder, PROMPT, RiMOM, S-Match, SAMBO).
  - The *Ontology localization and profiling* component adapts an ontology according to some language, context or user profile (e.g., LabelTranslator, lemon editor).
  - The *Ontology evaluator* evaluates ontologies, either their formal model or their content, in the different phases of their life cycle (e.g., CleOn, ConsVISor, Eyeball, VRP).
- **Instance generation**. This dimension includes components that generate ontology instances.
  - The *Instance editor* allows manually creating and modifying instances of concepts and of relations between them in existing ontologies (e.g., GATE, OCAT).
  - The *Manual annotation* component is in charge of manual and semi-automatic annotation of digital content documents (e.g. web pages) with concepts in the ontology. This annotation process may be assisted or guided by a machine (semi-automatic annotation) (e.g., GATE, OCAT, OntoMat, Magpie, M-OntoMat, PhotoStuff).

- The *Automatic annotation* component automatically annotates digital content (e.g. web pages) with concepts in the ontology (e.g., KIM, GATE-ML).
- The *Ontology populator* automatically generates new instances in a given ontology from a data source (e.g., CLIE, NOR_2O, R_2O & ODEMapster, geometry2rdf).
- The *Instance matcher* automatically is in charge of manual and semi-automatic matching of instances from different ontologies (e.g., SILK, LIMES).

Regarding the ontology engineering dimension, of special relevance are those software platforms that cover more that one of the aforementioned components and that support most of the activities in the ontology development process. In this paper, we focus on the new generation of ontology engineering environments, particularly, on NeOn Toolkit, Protégé, and TopBraid Composer. They have extensible, component-based architectures, where new modules can easily be added to provide more functionality to the environment.

- **The NeOn Toolkit** (http://neon-toolkit.org/) is an ontology engineering environment that supports the complete life cycle of large-scale ontology networks. In order to support such a broad ontology modelling functionality, it has an open and modular architecture, which the NeOn Toolkit inherits from its underlying platform, Eclipse. Eclipse is a very rich development environment, which is widely adopted in the programming world and which perfectly fits to the modelling paradigm for ontologies. It provides developers with a framework to easily create, publish and integrate new features into the NeOn Toolkit. A substantial number of so-called plug-ins has been developed within and outside the NeOn consortium and are available at NeOn Toolkit homepage.

The NeOn Toolkit is available as an installable core version with the basic ontology functionality such as editing, browsing, ontology and project management. Currently, the following versions are available:

- The basic NeOn Toolkit provides the core functionality for handling OWL 2 ontologies.
- The NeOn Toolkit extended configuration includes advanced functionality for managing rule based models and ontology mapping facilities based on commercial extensions.

- **Protégé** (http://protege.stanford.edu/) is an open platform for ontology modelling and knowledge acquisition. It is an open source, standalone application with an extensible architecture. The core of this environment is the ontology editor, and it holds a library of modules that can be plugged, called plug-ins, to add more functions to the environment.

The main Protégé functions are to: load and save OWL and RDF ontologies; edit and visualize classes, properties, and SWRL rules; define logical class characteristics as OWL expressions; execute reasoners such as description logic classifiers; and edit OWL individuals for Semantic Web markup.
Protégé is available in different versions, each including different plug-ins, whose main difference is the ontology language that they support:

- Protégé version 3 supports OWL 1.0, RDF(S) and Frames.
- Protégé version 4 supports OWL 2.0.

- **TopBraid Composer** (http://www.topquadrant.com/products/TB_Composer.html) is a modelling environment for developing Semantic Web ontologies and building semantic applications. It is

fully compliant with W3C standards and offers support for developing, managing and testing configurations of knowledge models and their instance knowledge bases. It is implemented as an Eclipse plug-in.

TopBraid Composer incorporates a flexible and extensible framework with a published API for developing semantic client/server or browser-based solutions that can integrate disparate applications and data sources.

TopBraid Composer is available in three different versions: Free Edition, Standard Edition and Maestro Edition.

1.1.6. Ontology Engineering in Linked Data: How to publish data

Publishing Linked Data is a process that involves a high number of steps, design decisions as well as a wide range of technologies. Although some initial guidelines have been already provided by Linked Data publishers, these are still far from covering all the steps that are necessary (from data source selection to its publication) or giving enough details about all the steps. In this section, we summarize a set of methodological guidelines for the activities involved in the Linked Data publishing process. These guidelines consist of the following activities: (1) identification of the data sources; (2) vocabulary modelling; (3) generation of the RDF data; (4) publication of the RDF data; and (5) linking the RDF data with other datasets in the cloud.

1. *Identification of the data sources*. Within this activity, we identify and select the datasets that we want to publish. This is normally a costly and tedious activity that may require contacting the data owners, and government bodies. If lucky, we may have that data already available in a public data catalogue. A representative example of these catalogues in the Spanish context is the Aporta project (http://www.aporta.es) catalogue. Other possibility is to get an agreement with a particular government body to publish its datasets.

In the case of GeoLinkedData (http://geo.linkeddata.es) we have followed those two paths. In one hand, we have searched for open government information at the Spanish Statistical Institute (INE) open catalogue (http://www.ine.es). In the other hand, we have got an agreement with the Spanish Geographic Institute (IGN) for publishing its geospatial datasets.

2. *Vocabulary modelling*. After the identification and selection of the datasets we need to determine the ontologies to be used to model the data contained in those datasets. The most important recommendation in this context is to reuse as much as possible available ontologies that model the information needed. If we do not find any particular ontology suitable for our needs, we should create them, either from scratch or by reusing existing resources. This activity is well described in ontology engineering methodologies, for example the NeOn Methodology (Suárez-Figueroa, 2010), summarized in Section 3; and tools such as the NeOn Toolkit (presented in Section 5) can be used.

In the case of GeoLinkedData, our chosen datasets contain information such as time, administrative boundaries, unemployment, etc. For modelling the information contained in the datasets we have created an ontology network (Suárez-Figueroa, 2010). The vocabulary that models the information contained in the datasets has been developed by reusing the following available vocabularies or

ontologies: Statistical Core Vocabulary (SCOVO), FAO Geopolitical Ontology, hydrOntology, WSG84 Vocabulary, and Time Ontology.

3. *Generation of the RDF data*. The preliminary guidelines proposed in this chapter consider only the transformation of the whole data source content into RDF, i.e., following an Extract, Transform, and Load ETL-like process, by using a set of RDF-izers, i.e. ontology population tools. The guidelines are based on the method proposed in (Villazón-Terrazas et al., 2010) that provides guide for transforming the content of a given resource into RDF instances. The requirements of the transformation are (1) full conversion, this implies that all queries that are possible on the original source should also be possible on the RDF version; and (2) the RDF instances generated should reflect the target ontology structure as closely as possible, in other words, the RDF instances must conform to the already existing ontology schema.

In GeoLinkedData, given the different formats in which the selected datasets were available, we used three different RDF-izers for the conversion of data into RDF. We have used NOR_2O for transforming the spreadsheets, R_2O & ODEMapster for the databases, and geometry2rdf for the geospatial information.

4. *Publication of the RDF data*. The preliminary guidelines proposed here consider that we will serve RDF data from a particular ontology repository. Ideally, every RDF triple store software would provide a Linked Data interface. Using this interface, the administrator of the store would configure which part of the store's content should be made accessible as Linked Data on the Web.

In GeoLinkedData, for the publication of the RDF data we relied on Virtuoso Universal Server. On top of it, Pubby (http://www4.wiwiss.fu-berlin.de/pubby/) was used for the visualization and navigation of the raw RDF data. On top of these two systems, we have developed a web based application, map4rdf34, to enhance the visualization of the aggregated information. This interface combines the faceted browsing paradigm with map-based visualization using the Google Maps API.

5. *Linking the RDF data*. Following the fourth Linked Data Principle ("Include links to other URIs, so that they can discover more things"), the next activity is to create links between our RDF data set and external datasets. This activity involves the discovery of relationships between data items. We can create these links manually, which is a time consuming activity, or we can rely on automatic or supervised tools, such as SILK (http://www4.wiwiss.fu-berlin.de/bizer/silk/) or LIMES (http://aksw.org/Projects/LIMES). The activity consists in the following tasks: (a) to identify data sets that may be suitable as linking targets, (b) to discover relationships between data items of our data set and the items of the identified data sets in the previous task, and (c) to validate the relationships that have been discovered.

In the context of GeoLinkedData, we have identified as initial data sets to link with DBpedia (http://dbpedia.org) and Geonames (http://geonames.org), because these data sets include similar topics.

Conclusions

At the beginning of the 90's, ontology development was similar to an art: ontology developers did not have guidelines on how to build ontologies. Work on principles, methods and methodologies, together with supporting technologies and languages, made ontology development become an engineering discipline, the so-called Ontology Engineering.

In this paper, we have provided a general summary on this discipline, focusing on ontology definition and ontology components (classes, relations, axioms, and instances), and methodologies for building ontologies as well as languages and tools for ontology building.

At this moment, ontology engineers and practitioners have at their disposal different methodologies for ontology development. The classical ones (METHONTOLOGY, On-To-Knowledge, and DILIGENT) that provides a rigid workflow for building ontologies, and the new one, the NeOn Methodology that conducts developers along different scenarios and activities for which prescriptive guidelines are provided.

With respect to languages, the decision of which one(s) to use for implementing the ontology should be based on the needs in terms of expressiveness and reasoning. In this paper, we provide the list of the most commonly used languages with their key features.

Finally, ontology engineers and practitioners needs tools that help them carry out different activities of the ontology development process (such as, implementation, evaluation, ontology search). In the last years, the number of ontology tools has greatly increased and they can be grouped into seven different dimensions (data and metadata management, querying and reasoning, ontology engineering, ontology customisation, ontology evolution, ontology instance generation, semantic web services). In this paper, we provide a brief description of three ontology development environment (the NeOn Toolkit, Protégé, and TopBraid Composer) that can be included in the ontology engineering dimension.

To put together methodologies, languages, and tools and the Linked Data initiative, we have also presented some guidelines on how datasets should be published in the Web of Data.

References

[1] Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (2002) "Description Logic Handbook". Cambridge University Press.

[2] Brickely, D. and Guha, R.V. (eds.) (2004) "RDF Vocabulary Description Language 1.0: RDF Schema". W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-schema/

[3] Dean, M. and Schreiber, G. (eds.) (2004) "OWL Web Ontology Language Reference". W3C Recommendation 10 February 2004. http://www.w3.org/TR/owl-ref/

[4] Espinoza, M., Montiel-Ponsoda, E., Gómez-Pérez, A. (2009) "Ontology Localization". *The Fifth International Conference on Knowledge Capture (KCAP 2009)*.

[5] García-Castro, R., Muñoz-García, O., Gómez-Pérez, A., Nixon, L. (2009) "Towards a component-based framework for developing Semantic Web applications". *3rd Asian Semantic Web Conference (ASWC 2008)*. 2-5 February, 2009. Bangkok, Thailand.

[6] Gómez-Pérez, A., Fernández-López, M., Corcho, O. (2003) "Ontological Engineering". Springer Verlag. Advanced Information and Knowledge Processing series. ISBN 1-85233-551-3. November 2003.

[7] Gruber, T.R. (1993) "A translation approach to portable ontology specification". *Knowledge Acquisition* 5(2): 199-220.

[8] vanHarmelen, F., Patel-Schneider, P.F., Horrocks, I. (eds.) (2001) "Reference Description of the DAML+OILOntologyMarkupLanguage".http://www.daml.org/2001/03/reference.html. Technical report.

[9] Klyne, G., and Carrol, J. (eds.) (2004) "Resource Description Framework (RDF) Concepts and Abstract Syntax". W3C Recommendation 10 February 2004. http://www.w3.org/TR/rdf-concepts/

[10] Motik, B., Patel-Schneider, P.F. Parsia, B. (eds.) (2009) "OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax". W3C Recommendation 27 October 2009. http://www.w3.org/TR/owl2-syntax/

[11] Pinto, H. S., Tempich, C., Staab, S. (2004) "DILIGENT: Towards a fine-grained methodology for DIstributed, Loosely-controlled and evolvInG Engineering of oNTologies". In Ramón López de Mantaras and LorenzaSaitta, *Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 2004)*, August 22nd - 27th, pp. 393--397. IOS Press, Valencia, Spain, August 2004. ISBN: 1-58603-452-9. ISSN: 0922-6389.

[12] Prud'hommeaux, E. and Seaborne, A. (eds.) (2008) "SPARQLQuery Language for RDF" W3C Recommendation 15 January 2008. http://www.w3.org/TR/rdf-sparql-query/

[13] Staab, S., Schnurr, H.P., Studer, R., Sure, Y. (2001) "Knowledge Processes and Ontologies". *IEEE Intelligent Systems* 16(1):26–34.

[14] Studer, R., Benjamins, V. R., Fensel, D. (1998) "Knowledge Engineering: Principles and Methods". *Data & Knowledge Engineering* (25). Pages: 161-197.

[15] Suárez-Figueroa, M.C. (2010) "NeOn Methodology for Building Ontology Networks: Specification, Scheduling and Reuse". *PhD Thesis*, Spain. Universidad Politécnica de Madrid. June 2010. http://oa.upm.es/3879/

[16] Suárez-Figueroa, M.C., Gómez-Pérez, A., Muñoz, O., Vigo, M. (2010) "gOntt, a Tool for Scheduling and Executing Ontology Development Projects". *The 22nd International Conference on Software Engineering and Knowledge Engineering (SEKE 2010)*. San Francisco Bay, USA. July 1 - July 3, 2010.

[17] Suárez-Figueroa, M.C., Gómez-Pérez, A., Villazón-Terrazas, B. (2009) "How to write and use the Ontology Requirements Specification Document". *Proceedings of the 8th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2009)*.Vilamoura, Algarve-Portugal. 3-5 November 2009.

[18] Villazón-Terrazas, B., Suárez-Figueroa, M.C., Gómez-Pérez, A. (2010) "A Pattern-Based Method for Re-Engineering Non-Ontological Resources into Ontologies". *International Journal on Semantic Web and Information Systems* 6 (4) 27–63.