University of Belgrade
Faculty of Mathematics

Using Web Tools for Constructing an Ontology of Different Natural Languages

A Ph. D. Dissertation submitted to
The Dept. of Computer Science
Faculty of Mathematics
University of Belgrade

by

Emhimed Alatrash

Under the supervision of

Prof. Dr. Dušan D. Tošić

2013

Abstract

Ontologies, often defined as an explicit specification of conceptualization, are necessary for knowledge representation and knowledge exchange. This means that ontology describes concepts and relations that exist in a domain. To enable knowledge exchange, it is necessary to describe these concepts and relations in a better way than just ordering them in taxonomy. A computational ontology consists of a number of different components, such as Concepts, Instances, Individuals or Facts, Relations and Attributes.

The present research is intended to consider different software tools related to Semantic web, and achieve a kind of comparison among them. In fact, five ontology-editors are described and compared. They are: Apollo, Onto Studio, Protégé, Swoop and TopBraid Composer Free Edition. The structure and basic features of these editors as well as the way of using them are described. The main criterion used in the process of comparing these editors lies in their convenience for the user, and the possibility to apply them in different kinds of application.

The main goal of the work is to introduce a method for ontology construction of a certain domain in applying the Semantic web. A number of software tools adapted to build up the domain ontologies of most wide–spread natural languages are available; however accomplishing that for any given natural language presents a challenge. This research proposes a semi-automatic procedure to create ontologies for different natural languages. The approach utilizes various software tools that are available on the Internet, most notably DODDLE-OWL which is a domain ontology development tool implemented for English and Japanese languages. Through this tool, WordNet, Protégé and XSLT transformations, the researcher proposes a general procedure to construct domain ontology for any natural language.

Keywords: Web tools, natural languague processing, ontology, machine readablity, semantic analysis

Scientific field: Computer science

Specific scientific field: Computer linguistics

UDC number: 004.912: 004.77(043.3)

Acknowledgements

I would like to express the deepest appreciation to my supervisor prof. Dr. Dušan D. Tošić for his willingness to accept me as his student, his guidance, advice, warm encouragement, patience and frequent discussions while working on this research. I have learned a lot from him.

Special thanks are extended to my parents and my wife for their continuous precious support and patience over the four years to complete this work. Finally, I would like to express my gratitude to all my friends and colleagues who believe that I could make it this far.

Contents

List of Figures

List of Tables

Chapter 1

1.1 Introduction

Ontologies gain a lot of attention in recent years as tools for knowledge representation. However, in the context of information and computer science there are many definitions regarding "what is an ontology?" For example in [1] an ontology is defined as a formal and explicit specification of a shared conceptualization. The terms in the previous definition have the following meanings:

- **Formal** refers to the fact that an ontology should be machine-readable.
- **Explicit** means that the type of concepts used and the constraints on their using are explicitly defined.
- **Conceptualization** means that an abstract model of phenomena is identified by appropriate concepts to these phenomena.
- **Shared** reflects that ontology should capture consensual knowledge accepted by the communities.

Tim Berners Lee (2001) gives a more concrete definition of this concept. He defines an ontology as "a document or file that formally defines the relations among terms", underlying, in this way, the importance of the relational aspect (formally defined) between the elements composing the ontology. An ontology can simply be defined as a formal knowledge representation system (KRS) composed by three main elements: classes (concepts or topics), instances (which are individuals belonging to a class), and properties (which link classes and instances allowing to insert information regarding the world represented into the ontology).

Obtaining a structured representation of the information through the ontologies is one of the main objectives in order to realize the so called Semantic Web. Tim Berners Lee (1999) views the Semantic Web as an extension of the current web in which information is given in a well-defined meaning. That, according to his vision, should enable the machines to "understand" the semantics of the web resources and, therefore, to have a more "intelligent" behavior in their activities of search. In the context of the Semantic Web, in fact, ontologies are expected to play an important role in helping automated processes to access

information. Moreover, ontologies are expected to be used to provide structured vocabularies that explicate the relationships between different terms, allowing intelligent agents (and humans) to interpret their meaning.

Since the 2006 RDF[1], RDFs[2] and OWL[3] are generally considered as standard Semantic Web languages. An interpretation of some of the most known and widely used representation languages of ontologies will be given in detail in chapter 2.

Ontology development is a complex and largely domain-oriented process that can be benefited from as a tool support. Researchers have recently developed a lot of tools for developing ontology. Many ontology editors could be found on Internet. Some of them – like Apollo, OntoStudio, Protégé, Swoop and TopBraid Composer Free edition – are used by a big number of people**.**

Semantic Web has lately been a popular and prolific field of research with numerous scientific papers published on the topic so far. Ontology is an important component of the Semantic Web and a lot of papers about applying ontology in specific fields have been published (see [2] and [3]). At the same time, the production of software tools to support ontology and Semantic web has accelerated. A number of these tools are free and available on the Internet. Unfortunately, most of them are adapted to most widely used languages such as English, Spanish and French etc. Some natural languages are not as presented in these tools. It is a challenge to create domain ontologies for text written in these languages.

1.2 Problem description

The researcher is meant here to present some software tools related to Semantic web along with a comparison among these tools. In fact, five ontology-editors are described and compared. They are: Apollo, OntoStudio, Protégé, Swoop and TopBraid Composer Free Edition. The structure and basic features of these editors are described in addition to the way of using them. The main criterion utilized in the

[1] http://www.w3.org/RDF

[2] http://www.w3.org/TR/rdf-schema/

[3] http://www.w3.org/OWL

process of comparing these editors is represented in their convenience for the user and the possibility to apply them in different aspects of application.

The main goal of this piece of research is to present an approach for constructing ontologies for different natural languages. The idea is to combine different accessible software tools and unite them for the semi-automatic construction of Natural Language Ontologies (NLOs) through specific domains. This approach is designed to be general and applicable for any natural language.

1.3 Structure of the Thesis

The structure of the thesis is as follows:

Chapter 2: This chapter is devoted to overview of the basic concepts of ontologies and the semantic Web. Several languages related to ontologies are considered in this chapter.

Chapter 3: This chapter presents the development and comparison of ontology editors. The structure and basic features of these editors are described, as well as the way of using them.

Chapter 4: This chapter will focus on presenting the construction of ontology for some domain. This is a very important step in applying Semantic web. A semi-automatic procedure is proposed to do that. In this approach, different software tools available on Internet are used, moreover, the main role has DODDLE-OWL - a domain ontology development tool implemented for English and Japanese languages. By using this tool, WorNet and XSLT transformations, we propose a general procedure to construct domain ontology for any natural language.

Chapter 5: is a conclusion where we summarize the work done in the thesis along with the main contributions. It also provides a number of interesting future threads of investigation that are directly relate to this research.

Chapter 2

Basic Concepts of Ontologies and the Semantic Web

2.1 Introduction

In the last decades, the use of ontologies in information systems has become more and more popular in various fields, such as web technologies, database integration, multi agent systems, natural language processing, etc. Artificial intelligent researchers have initially borrowed the word "ontology" from Philosophy, then the word spread in many scientific domains and ontologies are now used in several developments.

Ontology is mainly created to describe its components which are: concepts (classes), instances, individuals or facts, attributes and relations.

In this chapter we describe the languages developed by the World Wide Web Consortium W3C[4] that are used to enable the Semantic Web and that are being used as a basis for the development of ontology languages for the semantic web.

A special attention will be dedicated to the following languages: XML, DAML + OIL, RDF, RDFs, OWL and F-Logic. OWL is divided into three more expressive sublanguages: OWL Lite, OWL DL, and OWL Full.

Tim Berners-Lee, creator of the World Wide Web, foresees a future when the Web will be more than just a collection of web pages ([4] and [5]). The Semantic Web is a vision for the future of the Web, in which information is given explicit meaning, making it easier for machines to automatically process and integrate information available on the Web. This will enable computer programs to perform complex tasks autonomously and to communicate amongst one another, and with humans, by being able to meaningfully interpret the wealth of knowledge that is available on the Web.

[4] http://www.w3.org.

2.2 Ontologies

The term ontology has been used in several disciplines, from philosophy, to knowledge engineering. In knowledge engineering ontology is comprised of concepts, concept properties, relationships between concepts and constraints. Ontologies are defined independently from the actual data and reflect a common understanding of the semantics of the domain of discourse. Ontology is an explicit specification of a representational vocabulary for a domain: definitions of classes, relations, functions, constraints and other objects. Pragmatically, a common ontology defines the vocabulary with which queries and assertions are exchanged among software entities. Ontologies are not limited to conservative definitions. Namely, in the traditional logic sense ontology only introduces terminology and does not add any knowledge about the world. To specify a conceptualization, we need to state axioms that put constraints on the possible interpretations for the defined terms [6].

2.3 What is an ontology?

"Ontology" is a term borrowed from philosophy that involves a branch of philosophy dealing with the nature and organization of reality. In the field of artificial intelligence, ontologies are not clearly defined. Ontologies are only considered in relation to different areas of knowledge. In fact, many definitions of ontologies are given, but the one that characterizes the essence of ontology is based on the definition relayed by Tom Gruber in [1] (similar could be found in [7]). "Ontology is a **formal** and **explicit** specification of a **shared conceptualization**." The meaning of the terms included in the definition is as follows:

- **Formal**: refers to the fact that an ontology must be understandable by the machine, that is, a machine must be able to interpret the semantics of the information provided;
- **Explicit**: means that the type of used concepts and the constraints on their using must be explicitly defined;
- **Conceptualization**: refers to an abstract model of some phenomenon in the world which identifies the relevant concepts of this phenomenon;

- **Shared**: indicates that the ontology knowledge supports consensus and it is not restricted to certain individuals, but accepted by a group.

Taxonomies, thesauri and ontologies confined to describe semantic relationships are like: "is-a-sort of" and its inverse "is represented by," or, more specifically, "is-a-subclass-of." More complex ontologies for the representation of more specific semantic links, for example, "is located-in". But above all, the most successful ontologies also allow the integration of properties, rules of using and restrictions.

2.4 Role of ontologies

Ontologies are a key enabling technology for the Semantic Web. They interweave human understanding of symbols with their machine process ability. Ontologies were developed in artificial intelligence to facilitate knowledge sharing and re-use. Since the early 1990s, ontologies have become a popular research topic. They have been studied by several artificial intelligence research communities, including knowledge engineering, natural-language processing and knowledge representation. More recently, the use of ontologies has also become widespread in fields such as intelligent information integration, cooperative information systems and information retrieval, electronic commerce and knowledge management.

The reason ontologies are becoming popular is largely due to what they promise: a shared and common under- standing of a domain that can be communicated between people and application systems. As such, the use of ontologies and supporting tools offers an opportunity to significantly improve knowledge management capabilities in large organizations. It describes Semantic Web-based knowledge management architecture and a suite of innovative tools for semantic information processing [34].

During experimentations, the methodologies for building ontology and adequate tools of development have emerged. Emerging from initial craft practices, a true engineering was formed around ontology, aiming for their construction, but more broadly their management throughout a life cycle. Ontology appears as a software component in information systems and providing them with semantic dimension which made them default to date.

The scope of ontology continues to broaden and covers consult systems (systems of decision support, systems for computer-based learning - e -learning - , etc.), problem solving and knowledge management systems (e.g. in the biomedical field). One of the biggest projects based on the use of ontology is to add a Web real layer of knowledge to finding information both at the syntactic level and at the semantic level.

2.5 The origin of ontology

The Knowledge Engineering (KE) is a branch of Artificial Intelligence (AI) issued of the study of Systems Experts (SE). If the latter were intended for the automatic resolution of problems, the knowledge-based systems (KBS). Allows: the storage and retrieval of knowledge, automated reasoning on the stored knowledge (unbiased type of reasoning), editing stored knowledge (adding or deleting knowledge) and, with the development of networks, knowledge sharing between computer systems.

Generally speaking, it is no longer problem about manipulating knowledge in the machine, which brings at the end the solution of the problem, but keeps room for cooperation between the system and the human user (systems of decision support, systems for computer-based learning, e-learning and information retrieval on the Web). The system must have access not only to the terms used by the user, but also must has the semantic that it associates with different terms, otherwise no effective communication is possible. More specifically, the symbolic representations used in machines should make sense for both: the machine and users, ("make sense" here means that one can link the information shown in other information).

For this, the representation of knowledge in the form of logical rules, for use in SE, is no longer sufficient. To model of semantic richness of knowledge, new formalisms, representing knowledge at the conceptual level, which include the "cognitive structure" of a domain, are introduced.

Language-based frames, description logics and conceptual graphs are examples of such formalisms. These languages allow representing the underlying concepts of

a domain of knowledge, relationships between them, and semantics of these relationships, regardless of how one wants to make use of this knowledge. Thus, the same knowledge base can be used in consultation or as a basis for reasoning. However, it should be noted that the conceptualization of a knowledge domain can not be ambiguous in the context of precise use. For example, a word can even have two different concepts in two different contexts of use [9]. We cannot conduct a completely independent way the representation of a domain of knowledge and model the treatments to be applied to them.

In other words, modeling knowledge can only be done in a given field of knowledge and for a given purpose. These conditions are necessary for unison of the associated semantics under the domain. However, some authors believe that ontology is, by nature, intended for reuse [10], and tend to build ontology whose semantics is independent from any operational objective. The ontological level and the primitives used to represent knowledge are no longer words of natural language, conceptual primitives, or logical predicates, but statements that give a sense of knowledge, with an interpretation constraint. Ontology is a representation of knowledge, containing the terms and statements that specify the semantics of a given area of knowledge in a given operational framework.

The term "ontological engineering" was proposed in [11] to designate a new research field aimed at building computer systems relying on content, not on the mechanisms of manipulation of information.

Before presenting the languages and tools available for manipulating ontology, we now specify what are the primitives used in ontology.

2.6 Components of an Ontology

Ontology contains a number of diverse components. The names of these components differ according to the used ontology language, and the philosophical persuasion or background of the authors. Despite this, their core components are largely shared among different types of ontologies. These components can be divided into two kinds: those that describe the entities of the domain, here called

concepts, individuals and relationships, and those which enable the use of the ontology or describe the ontology itself.

2.6.1 Concepts:

Concepts of ontology are generally organized in taxonomies. Sometimes the notion of ontology is somewhat diluted, in the sense that taxonomies are to be seen as full ontologies [12].

Concepts, also known as classes, are used in a broad sense. They can be abstract or concrete, elementary or composite, real or fictions. In short, a concept can be anything about which something is said, and, therefore, could also be the description of a task, function, action, strategy, or reasoning process, etc. [13].

Most ontology languages allow the author to define Concepts on the basis of the above characteristics. Moreover, some languages, such as OWL, allow a broader definition of Concepts based on their membership. For example, the Concept "members of the Beatles" could be defined as the set of "John, Paul, George and Ringo".

One Concept may be a sub-concept (also known as sub-class, or kind of) of another Concept. This means that if the Concept C_1 is a sub-concept of C, then any individual of type C_1 will also be an individual of type C. It is possible within an ontology to explicitly state that C_1 is a sub-concept of C; in some languages, including OWL, it is also possible to infer this [14]. Concepts may also share relationships with each other; these describe the way individuals of one Concept relate to the individuals of another.

2.6.2 Instances, individuals or facts

Instances, individuals or facts are terms used to represent elements in the domain. Instances [15] represent elements of a given concept. Facts [16] represent a relation which holds between elements. Individuals [17] refer to any element in the domain which is not a class (both instances and facts).

2.6.3 **Relationships**:

Relations represent a type of association between concepts of the domain. The majority of relations link two concepts, so they are called binary relations.

The most important relation in ontology is the subsumption relation which includes “is-super class-of, is-subclass-of” and others. This helps to decide which objects are members of class of objects. The is-a relationship creates hierarchical taxonomy; a tree like structure that clearly shows how objects are related to each other. Another type of relation is “part-of”, which represents how objects are combined to form composite objects.

2.6.4 Attributes

Attributes describe properties of concepts and instances. The authors distinguish two types of attributes: class attributes and instance attributes. The first type describes concepts and takes their values in the concept where they are defined. The other describes instances and takes their values in them.

Figure 2.1 represents a simple ontology also called lightweight ontology containing classes, taxonomical relations.

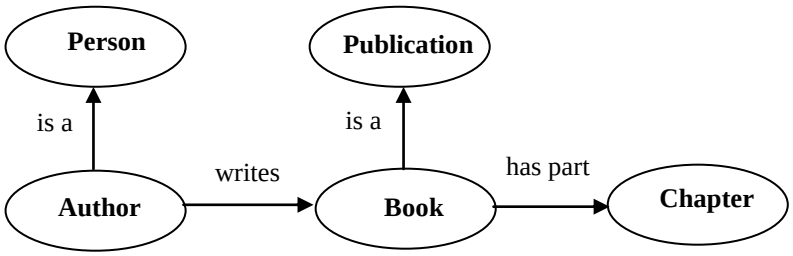

Figure 2.1 Example of a small ontology

2.7 Types of ontologies

There are different types of ontologies that support various uses of them. In literature, the classification is usually made using criteria that reflects the potential ontologies exhibited for reuse in engineering knowledge based system [26].

One of the most distinguished classifications of ontology has been provided by Guarino [27] and [28]. Figure 2.2 presents a summary of this classification. It distinguishes four types of ontologies classified according to their level of dependence on a particular task or point of view. These are as follows:

- Top-level ontologies: these ontologies describe very abstract and general concepts such as time and space in a manner that is not dependent on any particular problem on domain.

- Domain and task ontologies: these types of ontologies represent, respectively, knowledge about a specific domain (for example, health), or knowledge about a specific task (for example, scheduling). The representation is often achieved by specializing concepts introduced in top-level ontologies. This means that these ontologies, although narrower than top-level ontologies, offer more depth in their representation of the domain of discourse and a particular task.
- Application ontologies: these ontologies serve as a medium for describing concepts relating to a task in a particular domain. They often make use of both domain and task ontologies to describe, for example, roles played by domain entities in performing a specified task. As such, they are much narrower in scope than domain and task ontologies.

Figure 2.2 represents an inclusion hierarchy where the lower ontologies inherit and specialize concepts and relations from the upper ones. The upper ontologies tend to have a broader potential for reuse, than the lower ones, which have a much narrower scope.

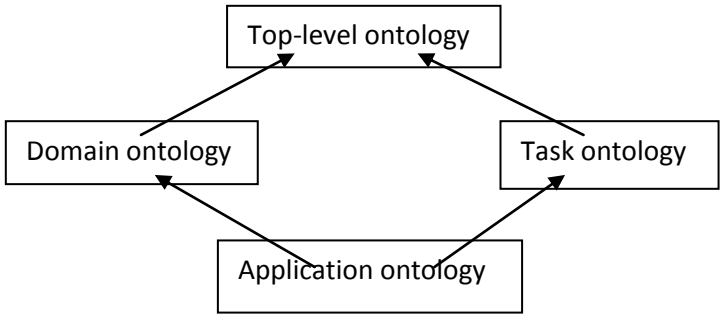

Figure.2. 3 Types of ontologies

2.8 Natural Language Processing (NLP):

It aims at acquiring, understanding and generating the human languages such as English, Arabic, Serbian, French, etc.

Natural language processing (`NLP) is a field of computer science, artificial intelligence, and linguistics concerned with the interactions between computers and

human (natural) languages. As such, NLP is related to the area of human–computer interaction. NLP encompasses anything a computer needs to understand natural language (typed or spoken) and also generate the natural language [30].

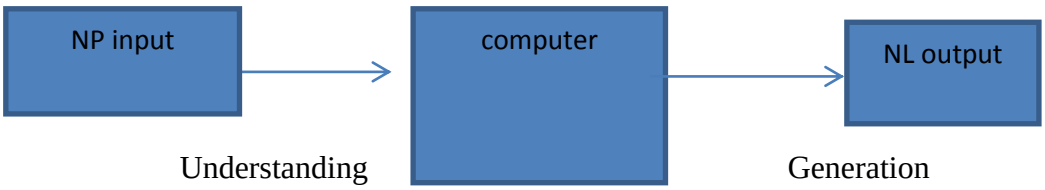

Figure2.3 natural language processing

2.8.1 Natural language understanding:

Communication between man and computer is one of the advantages of a natural language. Man already knows the natural language, so that he/she does not have to learn an artificial language or bear the burden of remembering its conventions over periods of disuse.

A language understanding program must have considerable knowledge about the structure of the language. This program must include a study of a language's words or vocabularies and how they combine into phrases and sentences. It must also provide an explanation of the meanings of words, and a way to discover the meaning of a sentence within its context. In addition, the program must show knowledge of the human's reason.

Natural language understanding consists of four main sections: Morphology, Syntax, Semantics and Pragmatics [30].

Morphology is the first stage of analysis once input has been received. It looks at the ways in which words break down into their components and how that affects their grammatical status.

Syntax involves applying the rules of the target language's grammar. Its task is to determine the role of each word in a sentence and organize this data into a structure that is more easily manipulated for further analysis.

Semantics is the examination of the meaning of words and sentences. Semantics convey useful information relevant to the scenario as a whole.

Pragmatics is the sequences of steps taken to expose the overall purpose of the statement being analyzed. This will be broken down into ambiguity and disambiguation to facilitate understanding.

2.8.2 Natural language generation

It is a sub-field of natural language processing that focuses on the generation of written texts in natural languages from some underlying non-linguistic representation of information, generally from databases or knowledge sources. Accomplishing this goal may be envisioned for a number of different purposes, including standardized and/or multi-lingual reports, summaries, machine translation, dialogue applications, and embedding in multi-media and hypertext environments. Consequently, NLG is associated with a large number of highly diverse tasks whose appropriate orchestration in high quality poses a variety of theoretical and practical problems. Relevant issues include: content selection, text organization and production of referring expressions, aggregation, lexicalization, surface realization, and coordination with other media, as well as. This process consists of text planning phase, sentence planning phase, and finally, text realization phase [30].

2.8.3 Steps in Natural Language Processing

Natural language processing is done at five steps. These steps are briefly given below.

1. **Morphological Analysis:** Individual words are analyzed to know their components and their non-word tokens. Punctuations, for instance, are separated from the words or sentences they identify.
2. **Syntactic Analysis:** Linear sequences of words are transformed into structures that show how the words relate each other. Some word sequences may be rejected if they violate a language's rules to show how words can be combined**.**
3. **Semantic Analysis:** Meanings are assigned to the structures created by the syntactic analyzer.
4. **Discourse Integration:** The meaning of an individual sentence may depend on the sentences that precede it and may influence the meanings of the sentence

(may depend on the sentences that precede it) that follows it.

Pragmatic Analysis: The structure representing what is said is reinterpreted to determine what is actually meant. For example, the sentence "Do you know what time is it?" should be interpreted as a request to tell the time.

2.9 Ontology Languages and Tools

There are several ontology languages like XML, RDF(S), DAML+OIL and OWL. Many ontology tools have been developed for implementing metadata of ontology using these languages.

2.9.1 Ontology languages

In this section, the researcher will focus on defining some of the most known and widely used representation languages of ontologies.

XML [20] provides syntax for structured documents, but imposes no semantic constraints on the meaning of documents. RDF is a data model for representing objects and relations between them. It provides simple semantics for the model and can be represented in XML syntax. RDF-Schema is a language for defining vocabulary for describing properties and classes of RDF resources. RDF(S) is used to define graphs of trio RDF, with semantics of generalization/prioritization of such properties and classes.

OWL adds vocabulary for describing properties and classes, relations between classes (e.g. disjointness), cardinality and characteristics of properties (e.g. symmetry). OWL is developed as an extension of RDF vocabularies, and it is derived from the ontology language DAML + OIL.

2.9.1.1 XML Extended Markup Language

XML is needed due to the limitations of hypertext markup language (HTML) and complexities of standard generalized markup language (SGML). It is an extensible markup language specified by the W3C, designed to make the interchange of structured documents over the Internet easier. The key to XML used to be document type definitions (DTDs), which define the role of each element of

text in a formal model. XML[5] is a language for describing data in a (semi-)structured manner. Data is described using a number of tags with arbitrary names that can contain other tags or arbitrary data. The names of the tags can be chosen arbitrarily by the creator of the xml document. XML can be used as a data exchange format, in which case all parties taking part in the exchange need to agree on a common structure for the XML document. Another possible use of XML, as it is done in the Semantic Web, is to use it as the serialization language for other languages. This has the advantage of the existence of many XML parsers, which can be reused for new languages, which use XML as their serialization language.

The structure of an XML document can be described using a Document Type Definition (DTD). A specific DTD describes the constraints on the structure of an XML document for it to be valid according to that DTD. A DTD describes the structure of a class of XML documents. Agreement on a DTD between different parties allows exchange of XML documents that conform to the DTD.

XML Schema[6] is a language used to define the structure of specific XML languages. XML Schema to restrict the structure of XML documents in well-defined ways and to provide the data types. It is a tool for defining grammars that characterize document trees (syntactic notion of validity). Through XML schemas, it is possible to constrain some meanings to the structure of a tree document.

XML Schemas and DTDs[7] both provide descriptions of document structures. The emphasis is on making those descriptions readable to automated processors such as parsers, editors, and other XML-based tools. They can also carry information for human consumption, describing what different elements should contain, how they should be used, and what interactions may take place between parts of a document. Although they use very different syntax to achieve this task, they both create documentation.

[5] http://www.w3.org/TR/2004/REC-xml11-20040204/

[6] http://www.w3.org/TR/xmlschema

[7] http://www.xml.com/pub/a/1999/12/dtd/

XML Schemas and DTDs provide a number of additional functions that make contributions to document content:

- Providing defaults for attributes: in addition to providing constraints on attribute content, DTDs and XML Schemas allow developers to specify default values that should be used if no value was set in the content explicitly.
- Entity declaration: DTDs and XML Schemas provide for the declaration of parsed entities, which can be referenced from within documents to include content.

Schemas and DTDs may also describe "notations" and "unparsed entities", adding information to documents that applications may use to interpret their content.

2.9.1.2 RDF and RDF Schema

RDF[8] (Resource Description Framework) is adopted by W3C as one of the standard formalism of knowledge representation on the Web. The syntax XML (Extended Markup Language) is already a standard. RDF is used to describe Web resources in terms of resources, properties and values. A resource can be a Web page (identified by its URI, United Resource Identifier) or part of a page (identified by a tag). The properties, covering the notions of attributes, relations and aspects, are used to describe a characteristic of a resource by specifying its value. Values can be resources or literals. RDF has formal semantics similar to those of conceptual graphs. It is virtually identical to a fragment of first order logic.

To describe any type of knowledge using this formalism, we first write the RDF semantic model. For example, to describe knowledge in terms of concepts and hierarchical relations, the introduction of types of "concepts" and "relations" and the properties of subsumption and instantiation is required. A database schema, including the semantic primitives commonly used, has been added to RDF and is

[8] http://www.w3.org/RDF

called the RDF Schema (RDF(S)[9]. Figure 2.4 shows the primitive of RDF (S). Concepts and relations are expressed in a document RDF (S) as instances of "Class" and "Property". Once when this pattern is stored on the Web, primitives described therein can be used in a page if we include a reference to the URI of the schema. An application requiring access to the semantics of the page then uses the scheme for interpretation. However, RDF(S) is not an operational language of representation, within the meaning because it does not allow the representation of the axioms and their use in reasoning.

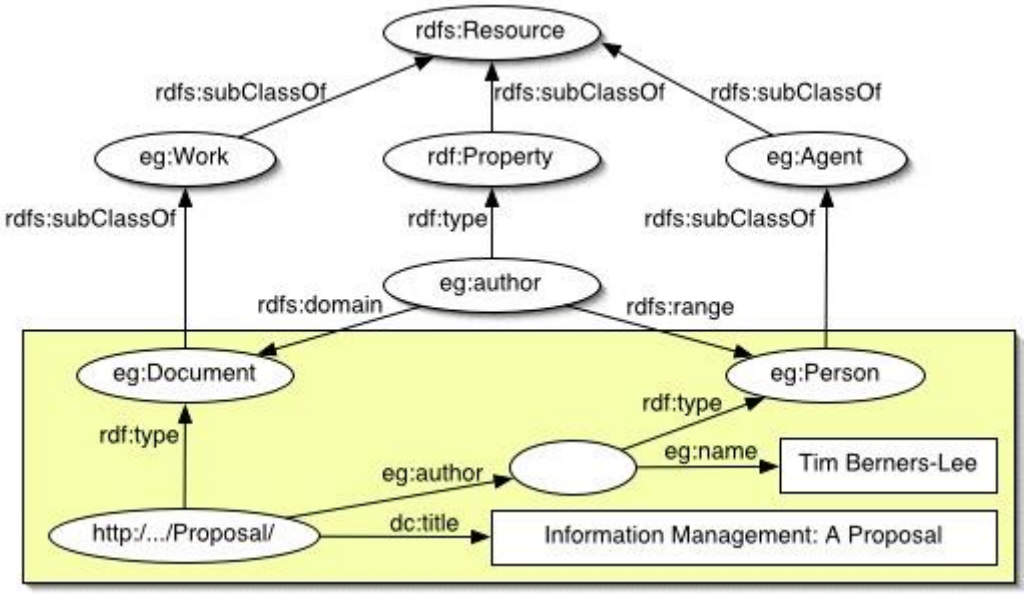

Figure.2. 4 Diagram of RDF (S)

2.9.1.3 DAML + OIL

OIL[10] (Ontology Interchange Language) is a Web-based representation and inference layer for ontologies, which combines the widely used modelling primitives from frame-based languages with the formal semantics and reasoning services provided by description logics. Furthermore, OIL is the first ontology representation language that is properly grounded in W3C standards such as RDF(S) and XML(S).

[9] http://www.w3.org/TR/rdf-schema/

[10] http://xml.coverpages.org/oil.html

It also allows refining the properties of RDF(S) binding cardinality or restricting the scope [18]. In OIL, an ontology is a structure made up of several components, organized in three layers: the object level (which deals with instances), the first meta level or ontology definition (which contains the ontology definitions) and the second meta level or ontology container (which contains information about features of the ontology, such as its author). Concepts, relations and functions and axioms can be defined in OIL.

DAML[11] (Darpa Agent Markup Language) is designed to allow the expression of ontologies in an extension of RDF. It offers the usual primitive representation frame-based and uses the RDF [The Darpa Agent Markup Language Homepage]. OIL language has merged to form the DAML+OIL[12]. DAML+OIL[13] is a semantic markup language for Web resources. It builds on earlier W3C standards such as RDF(S), and extends these languages with richer modelling primitives. DAML+OIL divide the universe into two disjoint parts. One part consists of the values that belong to XML Schema datatypes. This part is called the datatype domain. The other part consists of (individual) objects that are considered to be members of classes described within DAML+OIL (or RDF). This part is called the object domain.

2.9.1.4 OWL

The combination of RDF / RDF(S) and DAML + OIL has allowed the emergence of OWL (Web Ontology Language), a standard language for knowledge representation on the Web. Developed by the Working Group on the W3C Semantic Web, OWL can be used to explicitly represent the meaning of terms in vocabularies and the relationships between those terms. OWL also aims to make Web resources easily accessible to automated processes [19]. It is provided by structuring in an understandable and standardized manner, and by adding Meta information. To conclude, OWL has more powerful means to express the meaning and semantics

[11] http://www.daml.org/

[12] http://www.daml.org/2001/03/daml+oil-index.htm

[13] http://www.w3.org/TR/2001/NOTE-daml+oil-reference-20011218

than XML, RDF, and RDF(S). In addition, OWL reflects the appearance of diffuse sources of knowledge and allows information to be gathered from distributed sources, particularly for linking ontologies and import information to other ontologies.

2.9.1.5 Sublanguages of OWL

OWL has three expressive sub languages : OWL Lite, OWL DL, and OWL Full:

1. OWL Lite supports users primarily needing a hierarchy of classification and simple constraints (a limit is set to 0 or 1 element, for example). It has a lower formal complexity than OWL DL. OWL Lite supports only a subset construction of OWL.

2. OWL DL: According to its name, OWL DL uses the description logic DL [21]. It was defined for users who demand a maximum expressiveness while retaining information completeness (all conclusions are guaranteed to be computable) and the possibility of making a decision (calculations will end in a finite time). It includes all OWL language constructs, which may be used only under certain restrictions.

3. OWL Full: It was defined for users who want maximum expressiveness and RDF syntactic freedom, but with no guaranteed information. OWL Full allows an ontology that increases the significance of the known predefined vocabulary (RDF or OWL). OWL Full allows an ontology to augment the meaning of the pre-defined (RDF or OWL) vocabulary. It is unlikely that any reasoning software will be able to support every feature of OWL Full. In other words, using OWL Full compared OWL DL, reasoning support is less predictable since the full implementation of OWL Full does not currently exist.

OWL DL and OWL Full maintain the same set of OWL constructs. The difference lies in restrictions on the use of some of its features and characteristics of the use of RDF. OWL Lite allows free mixing of OWL with RDFS and, like RDFS, does not impose a strict separation of classes, properties, individuals, and data values.

OWL Full can be viewed as being an extension of RDF, while OWL Lite and OWL DL can be seen as extensions of a restricted view of RDF. So RDF users upgrading to OWL should be aware that OWL Lite is *not* simply an extension of RDF Schema. OWL Lite is a light version of OWL DL and puts constraints on the use of the RDF vocabulary (*e.g.*, disjointness of classes, properties, etc.). OWL Full is designed for maximal RDF compatibility and is therefore the natural place to start for RDF users. When opting for either OWL DL or OWL Lite one should consider whether the advantages of OWL DL/Lite (*e.g.*, reasoning support) outweigh the DL/Lite restrictions on the use of OWL and RDF constructs [21] and [22].

2.9.2 Ontology tools

Most of the languages are supported by different tools, but the researcher will not examine them all. Tools suitable for ontology development are editors like: Apollo, OntoStudio, Protégé, Swoop and TopBraid Composer Free edition. These tools are used by a huge number of people. They will be introduced and compared in the chapter 3.

2.10 Semantic Web

Tim Berners-Lee, James Hendler and Ora Lassila in [4] define semantic web as "an extension of the current web in which information is well-defined meaning given, enabling better computers and people to work in cooperation".

The web allows us today to access documents and services via Internet. The interface to access services is represented by web pages written in natural languages, understood by humans. The semantic web is an extension of the current web where information can give a well-defined meaning, providing both computers and people to work and cooperate effectively. The vision of the semantic web has been first introduced by Tim Berners-Lee in [4] and [5]. The utility of the semantic web can be seen through the following examples: a consumer is comparing prices of locally made sofas (i.e. postal code is the same), or checking online catalogs of various manufacturers of spare parts of a car brand (BMW). The answers to these questions can be found on the web, but not in a form usable by the machine. We always need

someone able to discern and give us the meaning of these responses and their importance to our needs. The semantic web approaches this problem in two ways. First, it allows local communities to insert their data so that the program will not be able to examine the format, images and advertisements in a web page to identify appropriate information. Then, it allows other people to write (generate) files that explain to machine the relationship among different data sets. For example, we can make a "semantic link" between the column "zip code" of a database and a form with a field "mail". This allows machines to follow links and facilitate the integration of data from multiple sources of information.

Tim Berners-Lee proposed a diagram (see [5]) showing the standard layers of the Semantic Web. It contains the following elements:

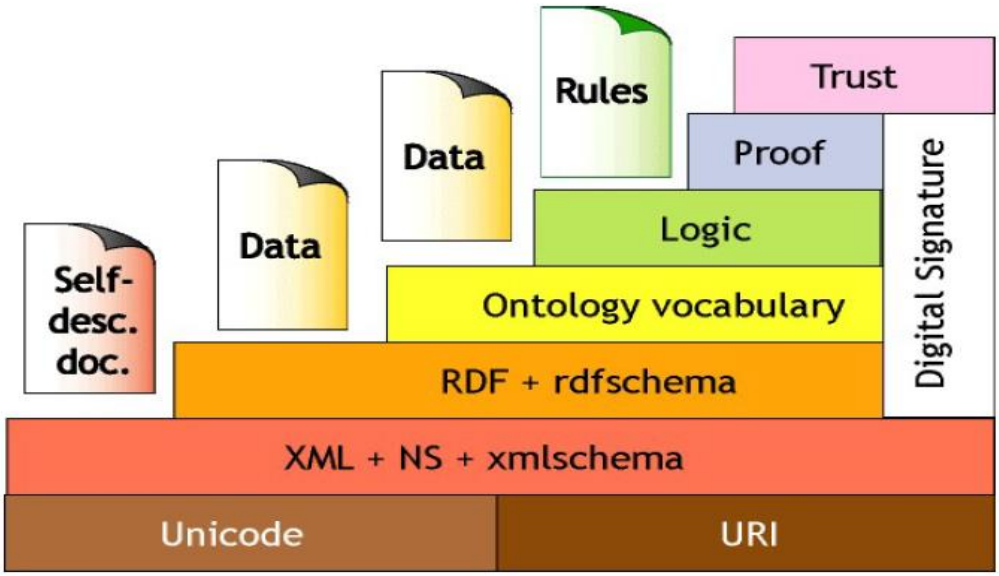

Figure.2. 5 Semantic Web Stack by Berners-Lee (2000)

2.10.1 The technologies used in the semantic web

A requirement for the functioning of the Semantic Web is the access of computer systems in structured collections of information and sets of logical rules that can be used to conduct automated conclusions. So it is vital to deal with the "representation of knowledge", a thematic concern in recent years in the field of Artificial Intelligence [29]. Among the technologies used by the Semantic Web are the following:

- URI (Uniform Resource Identifiers) is a fundamental component of the current Web. It provides a unique identification of resources and the relationships between these resources;
- XML (Extensible Markup Language) is a fundamental component for syntactic interoperability. It is a language developed by W3C and has been a Recommendation in 1999
- RDF (Resource Description Framework) is a means of encoding, exchanging and reusing of structured metadata. RDF does not specify the semantics of the resources described by the different user communities' metadata. Like XML, RDF is an extensible language, a meta-language. It is a part of "framework" for describing resources applicable to any application domain;
- The ontology layer provides more meta-information such as the cardinality of relationships, their transitivity, etc. ;
- The logical layer allows writing rules;
- The proof layer executes and evaluates the rules used;
- Layer trust that the Semantic Web can support. This component has not progressed far beyond a vision of allowing people to ask questions of the trustworthiness of the information on the Web, in order to provide an assurance of its quality.
- digital signatures are used to the detect the possible changes of documents

2.11 Conclusion

In this chapter we present the basic concepts of ontology and semantic web, which are the basis of our work. Here is a summary of the main points tackled in this chapter:

The semantic web is established as layers, and ontology is the core element of the Semantic Web. Ontology is an explicit and formal specification of a shared conceptualization. Ontologies can be classified according to their level of formalization. Designing an ontology is an alternative process. There are several

ontology specification languages with different characteristics such as DAML + OIL, RDF, RDFs, OWL and F-Logic.

At the end, we can say that the semantic web is not just like an individual web, but it is a wing of current web system in which the services, data and information are provided in distinct and precise meanings. In order to make the building of semantic web accessible, the W3C works actively on the classification of some open standards like the OWL and RDF.

Even though, the machines help managing the symbols according to the predefined procedures. Only the user of semantic web has the required interpretative capability to build and maintain the ontologies. The major advantage of the semantics is providing proper base for reasoning regarding the properties of schemes that perform the automatic translation of knowledge based upon ontology sharing. The developers are trying to create the services of semantic web enthusiastically.

Chapter 3

Comparison of Ontology Editors

3.1 Introduction

There are a lot of software tools related to Semantic web. Some significant Semantic web editors have designed various types of software for the creation and manipulation of ontologies. Many ontology editors could be found on Internet. Some of them, like Apollo, OntoStudio, Protégé, Swoop and TopBraid Composer Free Edition) are used by a huge number of people. For example, in [31], the Semantic web editor Protégé is classified as a "killer application". On the other hand, killer applications are defined as highly transformative technologies that create new markets and widespread patterns of behavior. Based on that, it becomes necessary to compare some of these editors.

Comparison could be done by using different criterions: generality, expressiveness, complexity, documentation, scalability etc. The main criterion is easiness to use and spreading of editors. The commonest editors probably used are: Apollo (see [41]), Protégé (see [44], [46], [36] and [47]), OntoStudio (see [42] and [43]), TopBraid Composer Free Edition (see [50], [52] and [51]) and Swoop (see [48], [49] and [36]). This is the main reason why it is necessary to compare these editors.

3.2 Ontology editor's development tools

The researcher will try to provide a broad overview of some available editors and environments that can be used for the building of ontologies. There will be also a brief description of each tool along with the group that has developed it. Comparison will be based on considering the different properties of editors. The researcher will take the following characteristics into account [40].

- **General description of the tools:** It includes information about developers and availability.
- **Software architecture and tool evolution:** It includes information about the tool architecture (standalone, client/server, n-tier application). It also explains how the tool can be extended with other functionalities/modules. Furthermore, it

describes how ontologies are stored (databases, text files, etc.), and show if there is any backup management system.

- **Interoperability with other ontology development tools and languages:** It includes information about the interoperability of the tool. Tool's interoperability with other ontology tools can be recognized by functionalities like (merging, annotation, storage, inference, etc.), in addition to translations to and from ontology languages.
- **Knowledge representation:** It is related to presenting the knowledge model of the tool. It also includes the possibility of providing any language for building axioms and whether tool gives support to methodology.
- **Inference services attached to the tool** tells if the tool has a built-in inference engine or it can use other attached inference engine. It also shows if the tool performs constraint/consistency checking. It also provides the possibility of classifying concepts automatically in concept taxonomy and capabilities to manage the exceptions in taxonomies.
- **Usability** shows the existence of the graphical editors for the creation of concept taxonomies and relations. It also shows the ability to prune these graphs and the possibility to perform zooms of parts of it. Moreover, it decides if the tool allows some kind of collaborative working and can provide libraries of ontologies.

The following candidates have been selected for comparison:

- Apollo
- OntoStudio
- Protégé
- Swoop
- TopBraid Composer Free Edition

All these tools are widespread in the ontology design and development sector. They are accepted by relatively large semantic web communities. These tools also provide the minimum necessary functionality supporting the ontology development process.

The ontology editors are tools that allow users to visually manipulate, inspect, browse, code ontologies, support the ontology development and

maintenance task. This part of the research is to provide a broad overview of some of the available ontology editor tools with a brief description of each tool. It presents the group that has developed it, its main features and functionalities, in addition to its URL, etc.

3.2.1 Apollo

Apollo [41] is a user-friendly knowledge modeling application. Apollo allows a user to model ontology with basic primitives such as classes, instances, functions, relations and so on. The internal model is a frame system based on the OKBC protocol. The knowledge base of Apollo consists of a hierarchical organization of ontologies. Ontologies can be inherited from other ontologies and can be used as if they were theirs by origin. Each ontology is the default ontology which includes all primitive classes. Each class can create a number of instances, and an instance inherits all slots of the class. Each slot consists of a set of facets. Apollo does not support graph view, web, information extraction and multi-user capabilities or collaborative processing. It features a strong type of consistency checking, store ontologies (XML files only) and import/export format (I/O plug-in architecture - export plug-ins to CLOS and OCML). Apollo is implemented in Java and it is available for a download from http://apollo.open.ac.uk/index.html.

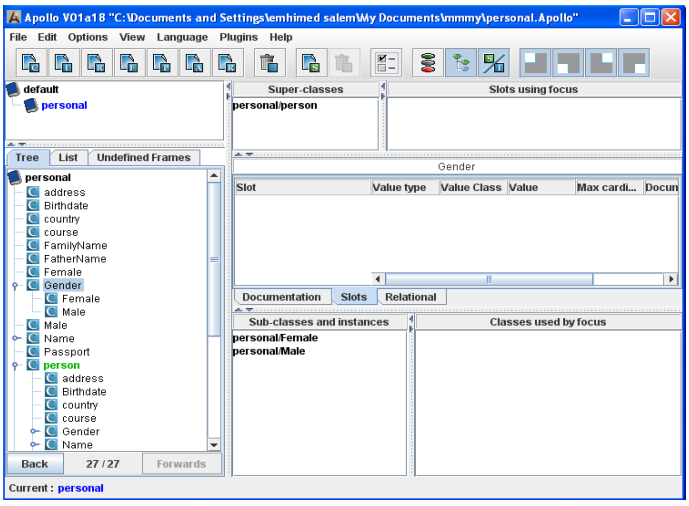

Figure.3. 6 Apollo screenshot

3.2.2 OntoStudio

OntoStudio is based on IBM Eclipse framework. It can be downloaded for three months free evaluation period. It is an Ontology Engineering Environment supporting the development and maintenance of ontologies by using graphical means. It is based on client/server architecture, where ontologies are managed in a central server. Various clients can access and modify these ontologies. An OntoStudio supports multilingual development, while a knowledge model is related to frame-based languages. It supports collaborative development of ontologies.

OntoStudio is built on top of a powerful internal ontology model. The tool allows the user to edit a hierarchy of concepts or classes. It is based on an open plug-in structure. The internal representation data model can be exported to DAML+OIL, F-Logic, RDF(S), and OXML. Additionally, ontologies can be exported to relational databases via JDBC. An OntoStudio can import external data representation in DAML+OIL, Excel, F-logic, RDF(S), database schemas (Oracle,MS-SQL, DB2,MySQL), and OXML. Moreover, it can import and export OWL files. It provides an API for accessing ontologies in an object-oriented fashion. The default API implementation stores ontologies in main-memory, but an additional API exists for persistent storage. The inference engine that an OntoStudio uses is an OntoBroker (An OntoBroker is the result of several years of investigation. It has become a commercial product, recently). Using this engine, OntoStudio exploits the strength of F-Logic and can represent expressive rules. An OntoStudio supports collaborative ontologies by using the OntoBroker Enhancement Collaborative server [42], [43].

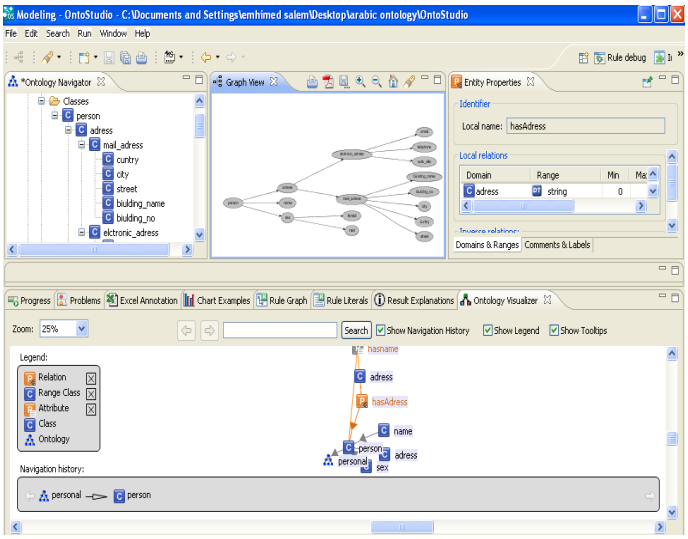

Figure.3. 7 OntoStudio editor Screenshot

3.2.3 Protégé ontology editor

Protégé is a free open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-base applications with ontologies. It implements a rich set of knowledge-modeling structures and action that support the creation, visualization and manipulation of ontologies in various representation formats. It can be customized to provide domain-friendly support for creating knowledge models and entering data. Also, it can be extended by a plug-in architecture and Java-based application programming interface (API) for building knowledge-base tools and applications. Protégé allows the definition of classes, class hierarchy's variables, variable-value restrictions, and the relationships among classes and the properties of these relationships. Protégé is freely available for download.

Stanford has a tutorial that covers the basics of using Protégé with the OWL plug-in. Protégé-OWL provides a reasoning API that can access an external DIG-compliant reason, enabling the inferences about classes and individuals in an ontology [44], [45]. Protégé includes an interface for SWRL (Semantic Web Rule Language), which sits on top of OWL to do math, temporal reasoning, and adds Prolog-type reasoning rules. The significant advantage of Protégé is its scalability and extensibility [36]. Protégé allows to build and to process large ontologies in an

efficient manner. Through its extensibility Protégé might be adopted and customized to suit users' requirements and needs. The most popular type of plug-ins is tab plug-ins. currently available tabs provide capabilities for advanced visualization, ontology merging, version management, inference, and so on. The OntoViz and Jambalaya tabs, for example, present different graphical views of a knowledge base, with the Jambalaya tab allowing interactive navigation, zooming in particular elements in the structure, and different layouts of nodes in a graph to highlight connections between clusters of data [46]. Protégé supports collaborative ontology editing as well as annotation of both ontology components and ontology changes [47].

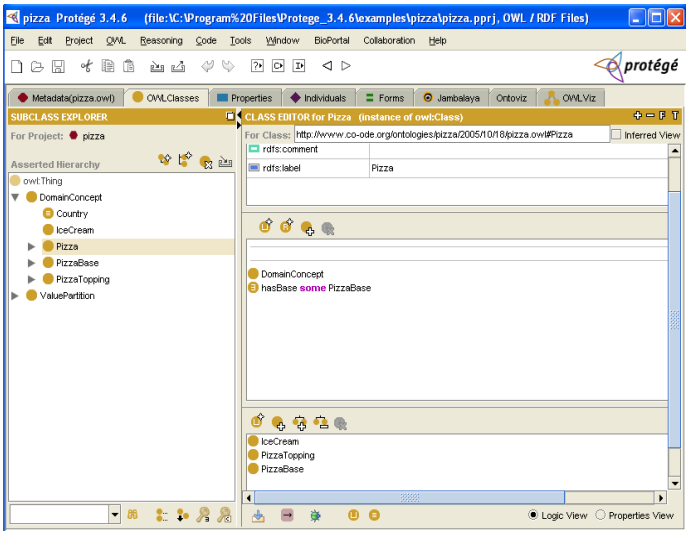

Figure.3. 8 Protégé 3.4 screenshot

3.2.4 Swoop

Swoop is an open-source, Web-based OWL ontology editor and browser [48]. Swoop contains OWL validation and offers various OWL presentation syntax views (Abstract Syntax, N3 etc). It has reasoning (RDFS-like and Pallet) support (OWL Inference Engine), and provides a Multiple Ontology environment, by which entities and relationships across various ontologies can be compared, edited and merged seamlessly. Different ontologies can be compared against their Description Logic-based definitions, associated properties and instances. Navigation could be simple and easy due to the hyperlinked capabilities in the interface of Swoop. Swoop does not follow a methodology for ontology construction. The users can

reuse external ontological data either by simply linking to the external entity, or by importing the entire external ontology. It is not possible to do partial imports of OWL, but it is possible to search concepts across multiple ontologies. Swoop uses ontology search algorithms that combine keywords with DL-based constructs to find related concepts in existing ontologies. This search is made along all the ontologies stored in the Swoop knowledge base [36]. Swoop has collaborative annotation with the Annotate plug-in [49].

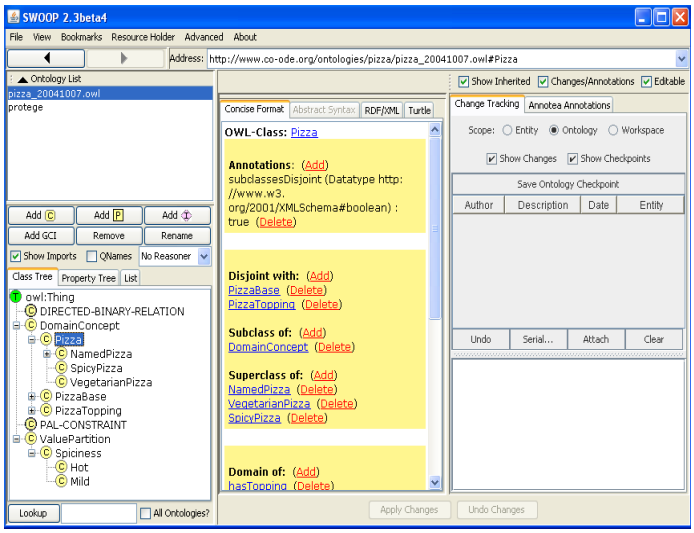

Figure.3. 9 Swoop: A Web Ontology Browser

3.2.5 TopBraid Composer Free Edition

TopBraid Composer comes in three editions:

**** Free Edition (FE)** which is an introductory version with only a core set of features,

**** Standard Edition (SE)** which includes all features of FE plus graphical viewers, import facilities, advanced refactoring support and much more, and

**** Maestro Edition (ME)** that includes all features of SE plus support for TopBraid Live, EVN and Ensemble as well as SPARQL Motion and many other power user features.

TopBraid Composer (FE), a component of TopBraid Suite, is a professional development tool for semantic models (ontologies). It is based on the Eclipse

platform and the Jena API. It is a complete editor for RDF(S) and OWL models, as well as a platform for other RDF-based components and services [50]. TopBraid Composer (FE) can load and save any OWL2 file in formats such as RDF/XML or Turtle [51].

TopBraid Composer (FE) supports various reasoning and consistency checking mechanisms. Consistency checking and debugging is supported by built-in OWL inference engine, SPARQL query engine and Rules engine. OWL description logic is supported via a range of built-in OWL DL engines such as OWLIM, Jena and Pellet [52]. The same composer (FE) also supports the SPARQL inference Notation (SPIN). On the other hand, SPIN can be used to define integrity constraints that can be used to highlight invalid data at edit time.

TopBraid Composer (FE) also provides inference explanation facilities for Pellet and SPIN. It can be used to connect RDF resources or ontologies with geospatial ontologies, and to query resources within a specific region. It can parse documents to extract RDF metadata from HTML pages. Therefore, the metadata can be treated like any other RDF source, and users can perform DL reasoning or SPARQL queries on it. TopBraid Composer (FE) may be used in a single user mode working with ontologies stored as files or in a database.

TopBraid Composer (FE) is a modeling tool for the creation and maintenance of ontologies. It provides: standard-based, syntax directed development of RDF(S) and OWL ontologies, SPARQL queries and Semantic Web rules import/export from a variety of data formats including RDF(S), XML, Excel, etc. Usability, extensibility and robustness are provided by underlying technologies – Eclipse and Jena [50]. TopBraid Composer (FE) can be downloaded and evaluated as a full version for a 30 days period.

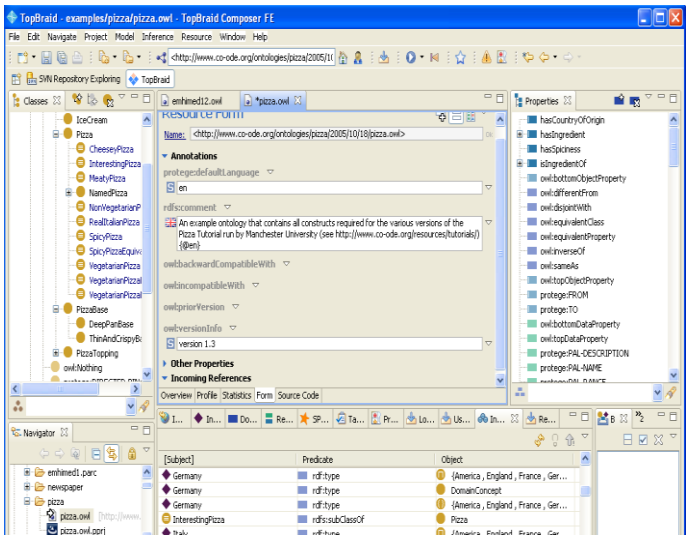

Figure.3. 10TopBraid Composer (FE) screenshot

3.3 Comparison of tools

The comments concerning this section are based on tools that have been described above. The tools are specified in alphabetical order: Apollo, OntoStudio, Protégé, Swoop and TopBraid Composer (FE). The results for comparison of tools are shown in the following:

Table3.6: General description of the tools

Feature	***Apollo***	***OntoStudio***	***Protégé***	***Swoop***	***TopBraid Composer (FE)***
Developers	KMI (Open University)	Ontoprise	SMI (Stanford University)	MND (University of Maryland)	Top Quadrant
Availability	Open source	Software License (Freeware)	Open source	Open source	Software license

General description of the tools (see Table 1) includes information about developers and availability. We can see from table 1 that: Apollo, Protégé and Swoop are open sources; and OntoStudio and TopBraid Composer (FE) are under the software license.

Table3.7: Software architecture and tool evolution

Feature	***Apollo***	***OntoStudio***	***Protégé***	***Swoop***	***TopBraid Composer (FE)***
Semantic web architecture	Standalone	Eclipse client/ server	Standalone and Client-server	Web-based and Client-server	Standalone Eclipse plug-in
Extensibility	Plug-ins	Plug-ins	Plug-ins	Yes Via plug-ins	Plug-ins
Backup management	No	No	No	No	No
Ontology storage	Files	DBMS	File and DBMS (JDBC)	As HTML Models	DBMS

Software architecture and tool evolution (Table 2) includes information about the necessary platforms to use the tool. In fact, the following information is provided: default architecture (standalone, client/server, n-tier application), extensibility, storage of the ontologies (databases, ASCII files, etc.) and backup management. All of these tools move towards Java platforms. Swoop is web based. However, Protégé, OntoStudio and Swoop have client/server architecture. Protégé, OntoStudio and TopBraid Composer (FE) use databases for storing ontologies.

Table3.8: Interoperability

Feature	***Apollo***	***OntoStudio***	***Protégé***	***Swoop***	***TopBraid Composer (FE)***
With other ontology tools	No	OntoAnnotate ,OntoBroker, OntoMat, Semantic and Miner	PROMPT, OKBC, JESS, FaCT and Jena	No	Sesame , Jena and Allegro Graph
Imports from languages	Apollo Meta language	XML(S), OWL, RDF(S), UML Diagram, database schemas (Oracle,MS-SQL, DB2,MySQL) , Outlook, file system Metadata and Remote OntoBroker	XML(S), RDF(S), OWL, (RDF,UML, XML)backend , Excel, BioPortal and DataMaster	OWL, XML, RDF and text formats	RDFa, WOL, RDF(s) ,XHTML, Microdata and RDFa Data sources, SPIN, News Feed, RDF Files into a new TDB, Email and Excel
Exports to languages	OCML and CLOS	XML(S), OWL, RDF(S),UML and OXML	XML(S), RDF(S), OWL, Clips, SWRL-IQ, Instance Selection, MetaAnalysis, OWLDoc, Queries and (RDF,UML, XML)backend	RDF (S), OIL and DAML	Merge / Convert RDF Graphs, RDF(S), WOL

Interoperability (Table 3) includes information about the tools interoperability with other ontology development tools and languages, translations to and from some ontology languages. It is another important feature in the integration of ontologies in applications. Most of these tools support import and export to and from many

languages in a variety of formats. TopBraid Composer (FE) supports the import of RDFa, WOL, RDF(s), XHTML, Microdata and RDFa Data sources, SPIN, News Feed, Email and Excel.

Swoop supports RDF (S), OIL and DAML. Apollo supports Apollo metalanguage, etc. Protégé supports the import of text files, database tables and RDF files. OntoStudio supports database schemas (Oracle, MS-SQL, DB2, MySQL), Outlook E-mails, etc. TopBraid Composer (FE) supports the export of Merge / Convert RDF Graphs, RDF(S) and WOL. Also Swoop supports RDF (S), OIL and DAML and Apollo supports Apollo metalanguage. Most of them support OWL, RDF(S) and XMl(S). However, there is no comparative study on the quality of each of these translators. Moreover, there are no experimental results about the possibility of exchanging ontologies between different tools and knowledge on the loose in the translation processes.

Table 9: Knowledge representation and methodological support

Feature	***Apollo***	***OntoStudio***	***Protégé***	***Swoop***	***TopBraid Composer (FE)***
KR paradigm of knowledge model	Frames (OKBC)	Frames and First Order logic	Frames, First Order logic , SWRL and Metaclasses	OWL	RDF, OWL and SWRL
Axiom language	Unrestricted	Yes (F-Logic)	Yes (PAL)	OWL-DL	OWL-DL
Methodological support	No	Yes (Onto-Knowledge)	No	No	No

From the **knowledge representation** point of view (**Table 4**), there is the family of tools which allows representation of knowledge following a hybrid approach based on frames and first order logic. Additionally, Protégé provides flexible modeling components like metaclasses. OntoStudio gives support to the Onto Knowledge methodology.

Table3.10: Inference services

Feature	*Apollo*	*OntoStudio*	*Protégé*	*Swoop*	*TopBraid Composer (FE)*
Built-in inference engine	No	Yes ontobroker	Yes (PAL)	No	WOL, SPARQL and Rule
Other attached inference engine	No	No	RACER, FACT, FACT++, F-logic and Pellet	Pellet and RDF-like	OWLIM, Pellet and SPARQL Rules
Constraint/ Consistency checking	Yes	Yes	Yes	Only with reasoner plug-in	Yes
Automatic classifications	No	No	No	No	No
Exception Handling	No	No	No	No	Yes

Inference services are presented in (**Table 5**)**.** This includes: built-in and other inference engines, constraint and consistency checking mechanisms, automatic classifications and exception handling, among others. For built-in inference engine Protégé uses PAL, OntoStudio uses OntoBroker and TopBraid Composer (FE) uses WOL, SPARQL and Rule. Protégé, Swoop and TopBraid Composer (FE) have external attached inference engines. TopBraid Composer (FE) uses the Exception Handling.

Table 11 : Usability of tools

Feature	*Apollo*	*OntoStudio*	*Protégé*	*Swoop*	*TopBraid Composer (FE)*
Graphical taxonomy	No	Yes	Yes	Yes	No
Graphical prunes (views)	No	Yes	Yes	No	No
Zooms	No	Yes	Yes	No	No
Collaborative working	No	Yes	Yes	Yes	Yes
Ontology libraries	Yes	Yes	Yes	No	Yes

Usability is related to Graphical editors, collaborative working and the provision of reusable ontology libraries. For most users, Protégé provides friendly possessing and easy to use graphical interface. Additionally, in Protégé and OntoStudio the layout of the interface and visualization of the ontology can be customised. Protégé and OntoStudio allow graphical taxonomy viewing, pruning and zooming. Help systems are essential to users and should be readily available and easy to use. The Apollo, Protégé, OntoStudio and TopBraid Composer (FE) help system is made up of a help on icons, tutorial, users guide. Swoop do not provide a help function in the user interface. Collaboration is essential in the process of building very large and extensive ontologies and Protégé, OntoStudio and TopBraid Composer (FE) allow collaborative construction of ontologies. Swoop allows users to write and share annotation on any ontological entity.

Conclusion

To sum up, the Apollo, Protégé 3.4 and Swoop tools are open source ontology and for the tools: OntoStudio and TopBraid composer (FE) software license is requested. Also, there are some tools application ontology demand learning / knowledge of a particular language Swoop. The tools: Protégé, TopBraid Composer (FE) and OntoStudio use databases for storing ontologies. The same applies to backup management functionality, which is just provided by TopBraid

Composer (FE). Protégé and OntoStudio are more graphical ontology tools. The Swoop is Web-based application. OntoStudio gives support to the Onto Knowledge methodology. TopBraid Composer (FE) uses the Exception Handling. Some of the tools only support the joint edition of the functions of browsing. Protégé, TopBraid Composer (FE), Swoop and OntoStudio editors provide documentation ontology, ontology import / export to different formats, graphical view of ontologies, ontology libraries and attached inference engines and Apollo supports Apollo metalanguage.

It is quite clear that Ontology development is mainly an ad-hoc approach. Among several viable alternatives, a user needs to find which one would work better for the projected task and which one easily and effectively can be maintained and expressed. The foundation of ontology is logic. It is a model of reality, at the same time. Hence, the concepts in the ontology must reflect this reality.

Chapter 4

Building ontologies for different natural languages

4.1 Introduction

Semantic Web has lately been a popular and prolific field of research with numerous scientific papers published on the topic so far. Ontology is an important component of the Semantic Web and a lot of papers about applying ontology in specific fields have been published (see [2], [3]). Ontologies are closely connected to Natural Language Processing (NLP) - a field of artificial intelligence, computer science and linguistics. As such, NLP is related to the area of human–computer interaction.

Ontologies provide an explicit and formal way for data interpretation, integration and sharing. It helps to understand natural (human) language. Understanding a natural language is not an aim per se, but it is useful in different fields such as: Information extraction (IE), Machine translation (MT), and Question answering (QA), etc. (Fig. 1.). Due to that, the production of software tools to support ontology and Semantic web has accelerated. A number of these tools are free and available on the Internet. The basic idea is to use available software tools and biding them by using programming techniques. The construction of ontologies from text focuses on the possibility to use a corpus of text, extract terms and relationships between words and then build ontology from these terms and relations. There is a lot of software tools for construction of ontologies from texts: Text2Onto (see: [53], [54] and [55]), Asium [56], TERMINAE [57] and DODDLE ([58], [59], [60] and [61]). A lot of these tools are free and available on the Internet.

Unfortunately, most of those tools are made to work with only a small set of most widely used languages such as: English, Spanish, French etc. Some natural languages are not as presented in these tools. It is a challenge to create domain ontologies for a text written in one of these languages.

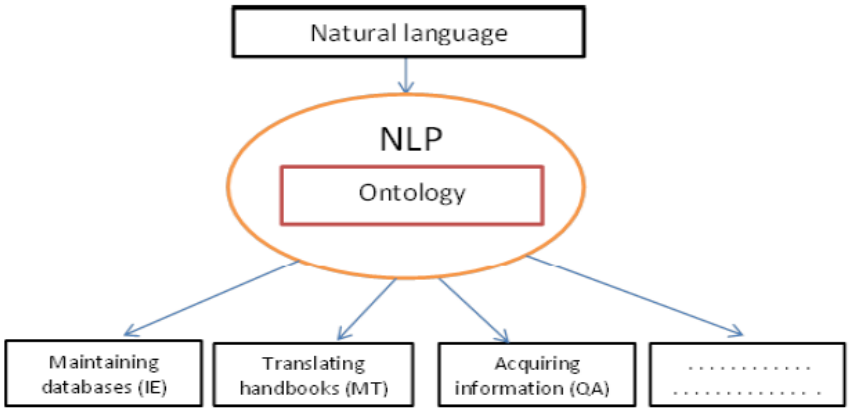

Figure.4.1. Relationship between natural language, NLP and ontology

The idea is to combine different accessible software tools for the purpose of semi-automatic construction of Natural Language Ontologies (NLOs) from specific domains. This approach aims to be general and applicable for any natural language. Understandably, certain adaptations become necessary according to the features of the used natural language. Nevertheless, the main tool in our approach is DODDLE-OWL. It supports the construction of both taxonomic and non-taxonomic relationships in ontologies. DODDLE is tailored for Japanese and English languages; it takes and uses Wordnet [62] as machine readable dictionary (MRD).

4.2 Semi-automatic creation of NLO

Even though automatic creation of domain NLO has been attempted, it is still a difficult task in general. It is particularly difficult to do so for the texts written in different natural languages and related to some domain. In this case the domain ontology structure depends, in some aspects, on human users. Because of that, it is convenient to provide refined semi-automatic software tool for building NLOs. That kind of tools is available on the Web and one of them is DODDLE-OWL (see: [60] and [61]). DODDLE-OWL is an interactive domain ontology development environment created for Japanese and English language. This environment is valid to be adopted by any natural language (that has dictionary on WordNet) by applying translation of original text into English text and transforming obtained English

ontology. Since DODDLE-OWL is an essential tool in our approach. A detailed description of DODDLE will be in the next few pages.

4.3 Ontology learning

Ontology learning (ontology extraction, ontology generation, or ontology acquisition) is a subtask of information extraction. The goal of ontology learning is to semi-automatically extract relevant concepts and relations from a given corpus or other kinds of data sets to form an ontology.

The automatic creation of ontologies is a task that involves many disciplines. Typically, the process starts by extracting terms and concepts or noun phrase from plain text using a method from terminology extraction. This usually involves linguistic processors (e.g. part of speech tagging, phrase chunking). Then, statistical or symbolic techniques are used to extract relation signatures. The intentional aspects of domain are formalized by ontology. Extensional part is commanded by the knowledge based on instances of concepts and relations on the basis of ontology.

4.4 Ontology Learning from Text

Currently, ontologies are mostly constructed by hand. It proves to be very ineffective and may cause a major barrier to their large-scale use in knowledge markup for the Semantic Web. Creating ambitious Semantic Web applications, based on ontological knowledge, implies the development of new highly adaptive and distributed ways of handling, and using knowledge enabling semi-automatic construction and refinement of ontologies. Automation of ontology construction can be implemented by a combined use of linguistic analysis and machine learning approaches for text mining. In this way the facilities for ontology construction and refinement (see e.g. [64] and the overview of ontology learning methods in [65]) are provided.

As human language is a primary mode of knowledge transfer, ontology development could be based more directly on the linguistic analysis of relevant documents. In recent years, the basic idea is to use available software tools and biding them by using programming techniques. The construction of ontologies from text focuses on the possibility to use a corpus of text, extract terms and relationships between words. This is to be followed by building an ontology from these terms and

relations. The commonest of these tools are adapted to the most wide–spread languages as: English, French, Spanish and so on. Moreover, some of them could be used for creating of ontologies for the other natural languages.

Text2Onto [53] is a framework for learning ontologies from textual resources. It is able to automatically create ontologies from a corpus of documents within a certain domain. Text2onto is a redesign of TextToOnto and is based on Probabilistic Ontology Model (POM) which stores the learned primitives independent of a specific Knowledge Representation (KR) language. It calculates a confidence for each learned object for better user interaction. It also updates the learned knowledge each time the corpus is changed and avoids processing it by scratch. It provides easy combination and execution of algorithms as well as writing new algorithms. Text2Onto uses the GATE architecture [66] to preprocess the text. It, also, depends on the output of GATE. POM (Probabilistic Ontology Model also called Preliminary Ontology Model) is the basic building block of Text2Onto. It is an extensible collection of modeling primitives for different types of ontology elements or axioms and uses confidence and relevance annotations for capturing uncertainty. It is KR language- independent and thus can be transformed into any reasonably expressive knowledge representation language such as OWL, RDFS and F-logic etc.
An important feature of Text2Onto is data-driven change discovery which prevents the whole corpus from being processed from scratch each time when it changed. If there are changes in the corpus, Text2Onto detects the changes and calculates POM deltas with respect to the changes. As POM is extensible, it modifies the POM without recalculating it for the whole document collection. The benefits of this feature are: the document reprocessing time is saved and the evolution of the ontology can be traced.

Text2Onto plug-in is a graphical front-end for Text2Onto that is available for the NeOn toolkit. It enables the integration of Text2Onto into a process of semi-automatic ontology engineering [54].

Text2Onto[14] provides full support for learning ontologies from English and Spanish corpora, as well as, partial support for ontology extraction from German texts.

Asium [56] is software of interactive learning used for the construction of hierarchies from texts. Asium uses a parser to extract subjects and complements of object. Using the associated verbs and the number of occurrences of the topics/supplements, Asium constructs base of classes including all levels and offers to user a means to validate grouping.

Terminae: The TERMINAE [57] tool is easy to install, however it requires the installation of further linguistic analysis tools (Syntex and Lexter). Although the interface is simple, it clearly requires previous natural language processing training or knowledge to give the appropriate orders for analysis.

Yoshikoder[15]: Yoshikoder is also easy to download and install, together with a .txt converter tool (Yoshikoder's input format). It supports multiple languages and the process for obtaining a report on word frequencies of a document, exportable to excel/html format is relatively straightforward. This report "lists the number of times each word type occurs and the corresponding proportion of the texts its tokens take up". For further content analysis (clusters, etc.), dictionaries (containing categories and patterns that identify them) may be created.

DODDLE-OWL: DODDLE-OWL (a Domain Ontology rapiD DeveLopment Environment - OWL extension) is a domain ontology development tool for the Semantic Web. It is written in Java language. According to [58], "DODDLE-OWL reuses existing ontologies such as WordNet and EDR as general ontologies to construct taxonomic relationships (defined as classes) and other relationships (defined as properties and their domains and ranges) for concepts". An initial

[14] http://neon-toolkit.org/wiki/1.x/Text2Onto

[15] http://yoshikoder.sourceforge.net/

concept hierarchy is constructed as a (is-a) hierarchy of terms. Here, it is assumed that there is one or more domain specific document, and that the user can select important terms needed to construct domain ontology.

DODDLE-OWL has the following six main modules: Ontology Selection Module, Input Module, Construction Module, Refinement Module, Visualization Module, and Translation Module. We assume that there is one or more domain specific document. It is also assumed that the user can select important terms needed to construct domain ontology (Figure.4.2, see [60] and [61]).

First, as an input to DODDLE-OWL, the user selects several concepts in Input Module. In Construction Module, DODDLE-OWL generates the basis of the ontology, an initial concept hierarchy and set of concept pairs, by referring to reference ontologies and documents. In Refinement Module, the initial ontology – generated by Construction Module — is refined by the user through interactive support by DODDLE-OWL.

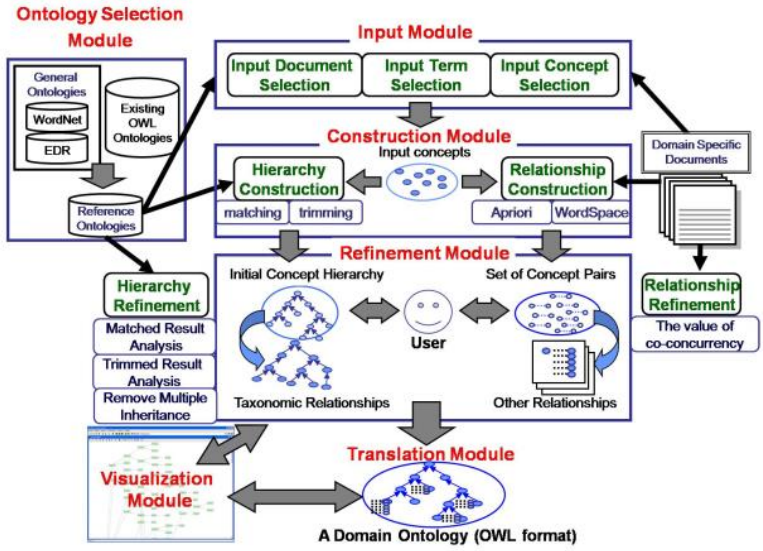

Figure.4.2 Overview of DODDLE-OWL

The ontology constructed by DODDLE-OWL can be exported with the representation of OWL. Finally, Visualization Module (MR3) (described in [67] and [59]) is connected with DODDLE-OWL and works with a graphical editor (see: [60] and [61]).

4.5 Construction of domain NLO for different languages

In order to construct the NLO for different languages, text document from any natural language is translated to English language.

We start to translate text document from Arabic language to English language by any text translation tools like Babylon and Google translation. For example, if we take this Arabic text:

" البرمجة هي فن جميل. البرمجة هي المقرر التعليمي في كلية الرياضيات.البرمجة هو الانضباط الفعلي للغاية ويتم تعلمه في العديد من الكليات في العالم. الآن يمكننا استخدام الكثير من لغات البرمجة الجديدة وجعل العديد من البرامج المختلفة."

We get the English text like this one:

"Programming is a beautiful art. Programming is a course in the Faculty of Mathematics. Programming is a discipline very effective and it is learned in many colleges in the world. Now we can use a lot of new programming languages and make many different programs."

English ontology is built by using DODDLE-OWL. In DODDLE-OWL the following steps are executed:

1. In the Ontology Selection Module, user selects reference ontologies WordNet, EDR (general vocabulary dictionary or technical terminology dictionary) and existing owl ontologies in the ontology selection as shown in Figure 4.3.
2. In the Input Document Selection Module, user selects domain specific documents described in English. At this step, some words in the documents are extracted. At the same step, user can select part of speech (POS) for extraction of words from the documents. For example, if noun or verb words are extracted, checkbox the "Noun" or "Verb" should be checked as shown in Figure.4.3.
3. In the Input Term Selection Module, list of extracted terms is formed. This list includes: compound words, part of speech (POS), Term Frequency (TF of term t in document d is defined as the number of times that t occurs in d), Inverse Document Frequency (IDF estimate the rarity of a term in the whole document collection - if a term occurs in all the documents of the collection,

its IDF is zero) and TF-IDF in the documents (TF-IDF is weight of a term - the product of its TF weight and its IDF weight). Domain specific documents contain many significant compound words. Therefore, accurate extraction of compound words is necessary to construct domain ontologies. At this step, while considering part of speech (POS), TF, and so on, the user selects input terms which are significant terms for the domain. For certain domains, important terms do not occur in the documents. In such case, the Input Term Selection Module has a function allowing the manual addition of important terms as input terms by the user. In order to prevent the leakage of the selection of input terms from the documents, the Input Term Selection Module maintains the relationships between the extracted terms and the terms in the documents as shown in Figure4.3.

4. In the Input Concept Selection Module, the user identifies the word sense of input terms to map those terms to the concepts in the reference ontologies selected with the Ontology Selection Module. A particular single term may have many word senses. Therefore, there may be many concepts corresponding to the word. The Input Concept Selection Module has a function enabling automatic word disambiguation. This function shows the list of concepts, ordered by some criteria, corresponding to the selected input term. Input term not corresponding to the labels of concepts in the reference ontologies is undefined term. The input terms are also undefined terms if the concept exists, but there are no appropriate concepts in the reference ontologies. The user defines the undefined terms manually in the refinement module as shown in Figure4.3.
5. The Hierarchy Construction Module automatically generates the basis of ontology, an initial concept hierarchy (by referring to reference ontologies) and documents. An initial concept hierarchy is constructed as taxonomic relationship.

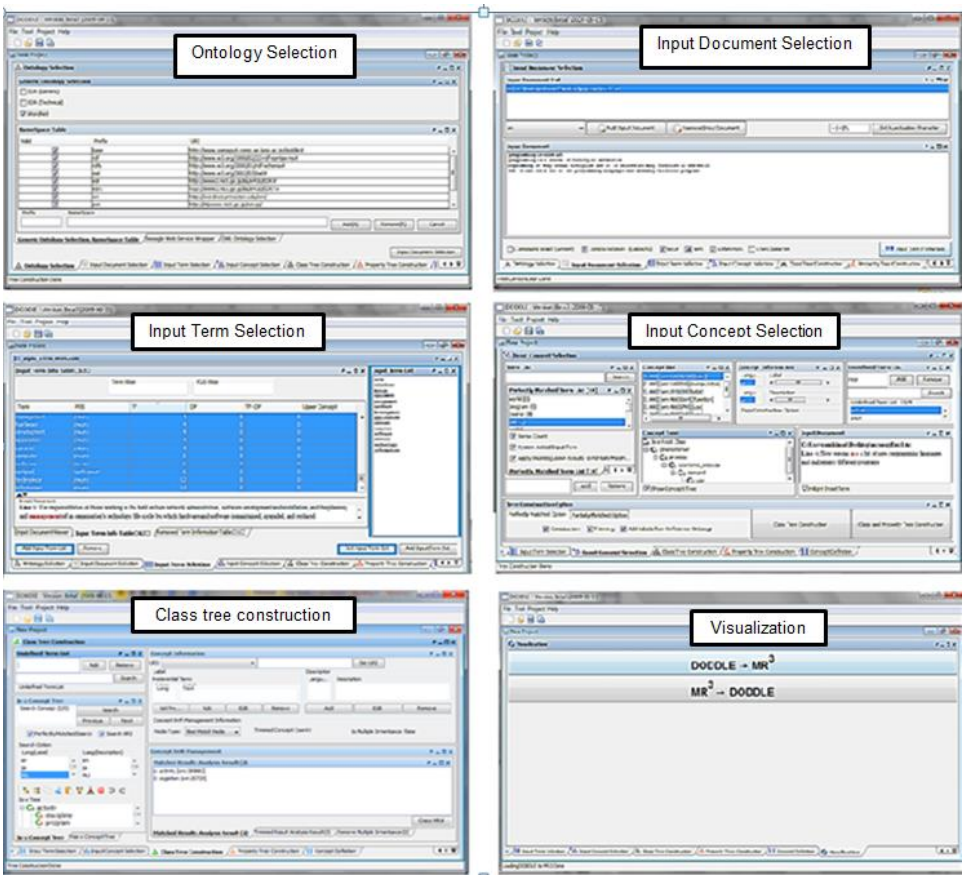

Figure.4.3 typical usage of DODDLE-OWL

6. DODDLE-OWL uses MR3: Meta-Model Management based on RDFs Revision Reflection [67] as the Visualization Module. Figure 4.4 shows the product of MR3 as: RDFs description and graphical representation. Finally, through the translation module, we can export constructed domain ontology described in RDFs. For example, a portion of the obtained English Ontology OWL code is presented in following document (1):

```
<rdf:RDF>
    <rdf:Description rdf:about="act">
    <rdfs:subClassOf rdf:resource="communication"/>
    <rdfs:comment xml:lang="en">photographs or other visual
representations in a printed publication; "the publisher was
responsible for all the artwork in the book"</rdfs:comment>
    <rdfs:label xml:lang="en">art</rdfs:label>
     </rdf:Description>
  <rdf:Description rdf:about="program">
```

```
    <rdfs:subClassOf rdf:resource="content"/>
    <rdfs:comment xml:lang="en">a series of steps to be
carried out or goals to be accomplished; "they drew up a six-
step plan"; "they discussed plans for a new bond
issue"</rdfs:comment>
    <rdfs:label xml:lang="en">program</rdfs:label>
     </rdf:Description>
  <rdf:Description rdf:about="discipline">
    <rdfs:subClassOf rdf:resource="content"/>
    <rdfs:comment xml:lang="en">a branch of knowledge; "in
what discipline is his doctorate?"; "teachers should be well
trained in their subject"; "anthropology is the study of human
beings"</rdfs:comment>
    <rdfs:label xml:lang="en">discipline</rdfs:label>
     </rdf:Description>
  <rdf:Description rdf:about="#CLASS_ROOT">
     </rdf:Description>
  <rdf:Description rdf:about="communication">
    <rdfs:subClassOf rdf:resource="abstraction"/>
    <rdfs:comment xml:lang="en">something that is communicated
by or to or between people or groups</rdfs:comment>
    <rdfs:label xml:lang="en">communication</rdfs:label>
     </rdf:Description>
  <rdf:Description rdf:about="activity">
    <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
    <rdfs:comment xml:lang="en">any specific activity; "they
avoided all recreational activity"</rdfs:comment>
    <rdfs:label xml:lang="en">activity</rdfs:label>
     </rdf:Description>
  <rdf:Description rdf:about="world">
    <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
    <rdfs:comment xml:lang="en">the concerns of the world as
distinguished from heaven and the afterlife; "they consider
the church to be independent of the world"</rdfs:comment>
    <rdfs:label xml:lang="en">world</rdfs:label>
     </rdf:Description>
  <rdf:Description rdf:about="course">
    <rdfs:subClassOf rdf:resource="activity"/>
```

```
    <rdfs:comment xml:lang="en">education imparted in a series
of lessons or class meetings; "he took a course in basket
weaving"; "flirting is not unknown in college
classes"</rdfs:comment>
    <rdfs:label xml:lang="en">course</rdfs:label>
     </rdf:Description>
   <rdf:Description rdf:about="faculty">
    <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
    <rdfs:comment xml:lang="en">the body of teachers and
administrators at a school; "the dean addressed the letter to
the entire staff of the university"</rdfs:comment>
    <rdfs:label xml:lang="en">faculty</rdfs:label>
     </rdf:Description>
  <rdf:Description rdf:about="abstraction">
    <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
    <rdfs:comment xml:lang="en">a general concept formed by
extracting common features from specific
examples</rdfs:comment>
    <rdfs:label xml:lang="en">abstraction</rdfs:label>
     </rdf:Description>
  <rdf:Description rdf:about="use">
    <rdfs:subClassOf rdf:resource="abstraction"/>
    <rdfs:comment xml:lang="en">what something is used for;
"the function of an auger is to bore holes"; "ballet is
beautiful but what use is it?"</rdfs:comment>
    <rdfs:label xml:lang="en">use</rdfs:label>
     </rdf:Description>
  <rdf:Description rdf:about="language">
    <rdfs:subClassOf rdf:resource="communication"/>
    <rdfs:comment xml:lang="en">the text of a popular song or
musical-comedy number; "his compositions always started with
the lyrics"; "he wrote both words and music"; "the song uses
colloquial language"</rdfs:comment>
    <rdfs:label xml:lang="en">language</rdfs:label>
     </rdf:Description>
   <rdf:Description rdf:about="content">
    <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
    <rdfs:comment xml:lang="en">the sum or range of what has
```

```
been perceived, discovered, or learned</rdfs:comment>
    <rdfs:label xml:lang="en">content</rdfs:label>
     </rdf:Description>
  <rdf:Description rdf:about="programming">
    <rdfs:subClassOf rdf:resource="activity"/>
    <rdfs:comment xml:lang="en">setting an order and time for
planned events</rdfs:comment>
    <rdfs:label xml:lang="en">programming</rdfs:label>
     </rdf:Description>
</rdf:RDF>
```

It is represented (by using MR[3]) in the following way:

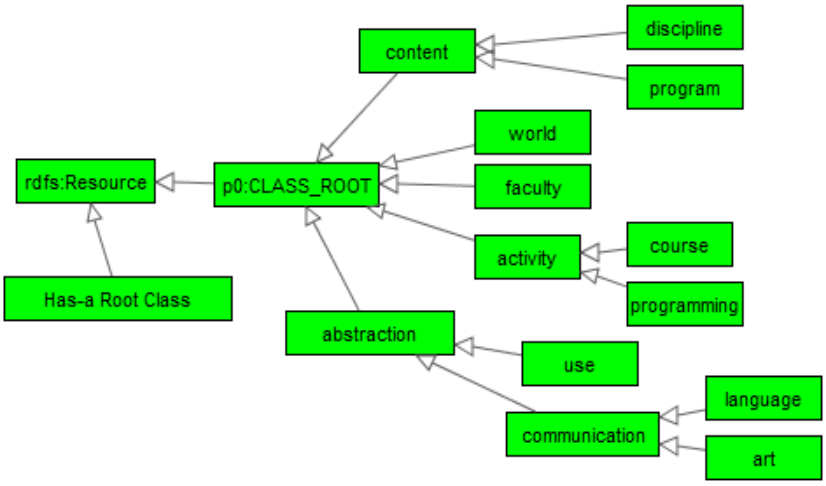

Figure.4.4 Graph for the English Ontology by using MR3

7. To build ontology represented by OWL, we use Protégé editor ([44]). Protégé has plugin to enhance ontology development such as the OWL plugin (see [45]). We use this possibility to get OWL document. For example, the document (1) is transformed in the following text.

```
<owl:Ontology rdf:about=""/>
  <owl:Class rdf:ID="course">
    <rdfs:subClassOf>
      <owl:Class rdf:ID="activity"/>
    </rdfs:subClassOf>
    <rdfs:label xml:lang="en">course</rdfs:label>
    <rdfs:comment xml:lang="en">education imparted in a series of
lessons or class meetings; "he took a course in basket weaving";
"flirting is not unknown in college classes"</rdfs:comment>
  </owl:Class>
```

```
  <owl:Class rdf:ID="CLASS_ROOT"/>
  <owl:Class rdf:ID="content">
     <rdfs:label xml:lang="en">content</rdfs:label>
    <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
    <rdfs:comment xml:lang="en">the sum or range of what has been
perceived, discovered, or learned</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="discipline">
    <rdfs:label xml:lang="en">discipline</rdfs:label>
<rdfs:subClassOf rdf:resource="#content"/>
    <rdfs:comment xml:lang="en">a branch of knowledge; "in what
discipline is his doctorate?"; "teachers should be well trained in
their subject"; "anthropology is the study of human
beings"</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="faculty">
<rdfs:comment xml:lang="en">the body of teachers and administrators
at a school; "the dean addressed the letter to the entire staff of
the university"</rdfs:comment>
    <rdfs:label xml:lang="en">faculty</rdfs:label>
    <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
  </owl:Class>
  <owl:Class rdf:ID="program">
    <rdfs:subClassOf rdf:resource="#content"/>
    <rdfs:comment xml:lang="en">a series of steps to be carried out
or goals to be accomplished; "they drew up a six-step plan"; "they
discussed plans for a new bond issue"</rdfs:comment>
    <rdfs:label xml:lang="en">program</rdfs:label>
  </owl:Class>
  <owl:Class rdf:ID="world">
<rdfs:comment xml:lang="en">the concerns of the world as
distinguished from heaven and the afterlife; "they consider the
church to be independent of the world"</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
    <rdfs:label xml:lang="en">world</rdfs:label>
  </owl:Class>
  <owl:Class rdf:ID="programming">
    <rdfs:label xml:lang="en">programming</rdfs:label>
```

```
    <rdfs:subClassOf>
      <owl:Class rdf:about="#activity"/>
    </rdfs:subClassOf>
<rdfs:comment xml:lang="en">setting an order and time for planned
events</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="language">
    <rdfs:label xml:lang="en">language</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="communication"/>
    </rdfs:subClassOf>
<rdfs:comment xml:lang="en">the text of a popular song or musical-
comedy number; "his compositions always started with the lyrics";
"he wrote both words and music"; "the song uses colloquial
language"</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="art">
    <rdfs:label xml:lang="en">art</rdfs:label>
    <rdfs:comment xml:lang="en">photographs or other visual
representations in a printed publication; "the publisher was
responsible for all the artwork in the book"</rdfs:comment>
<rdfs:subClassOf>
      <owl:Class rdf:about="#communication"/>
    </rdfs:subClassOf>
  </owl:Class>
  <owl:Class rdf:about="#communication">
    <rdfs:label xml:lang="en">communication</rdfs:label>
    <rdfs:subClassOf>
      <owl:Class rdf:ID="abstraction"/>
    </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">something that is communicated by or
to or between people or groups</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:about="#abstraction">
    <rdfs:label xml:lang="en">abstraction</rdfs:label>
    <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
    <rdfs:comment xml:lang="en">a general concept formed by
extracting common features from specific examples</rdfs:comment>
```

```
  </owl:Class>
  <owl:Class rdf:about="#activity">
    <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
    <rdfs:label xml:lang="en">activity</rdfs:label>
    <rdfs:comment xml:lang="en">any specific activity; "they avoided
all recreational activity"</rdfs:comment>
  </owl:Class>
  <owl:Class rdf:ID="use">
    <rdfs:comment xml:lang="en">what something is used for; "the
function of an auger is to bore holes"; "ballet is beautiful but
what use is it?"</rdfs:comment>
    <rdfs:label xml:lang="en">use</rdfs:label>
    <rdfs:subClassOf rdf:resource="#abstraction"/>
  </owl:Class>
  <owl:ObjectProperty rdf:ID="PROPERTY_HASA_ROOT"/>
  <owl:ObjectProperty rdf:ID="PROP_ROOT"/>
</rdf:RDF>
```

The similar ontology graph for the related English words could be generated as in Figure 4.5 by using Protégé editor like this:

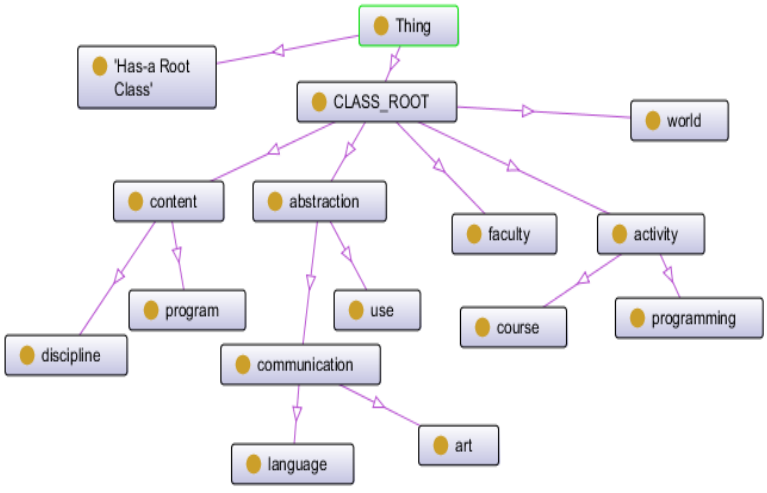

Figure 4.5 Ontology graph for English words

8- The most important step in the localization process is translation of ontology recognized in a source language into target language by using XSLT. The target language is Arabic (Modern Standard Arabic (MSA) is the language of today's Arabic newspapers, magazines, periodicals, letters and modern writers [68]). We are looking for tags <rdfs:label> , <owl:Class > (this includes rdf:ID and rdf:about) and

<rdfs:subClassOf> (this includes attributes rdf:about and rdf:resource) and duplicate them. For example, if the input document includes these tags:

```
<owl:Class rdf:ID="discipline">
    <rdfs:label xml:lang="en">discipline</rdfs:label>
    <rdfs:subClassOf rdf:resource="#content"/>
    <rdfs:comment xml:lang="en">a branch of knowledge; "in what
discipline is his doctorate?"; "teachers should be well trained in
their subject"; "anthropology is the study of human
beings"</rdfs:comment>
  </owl:Class>
```

The target language is Arabic (Modern Standard Arabic (MSA). We will get document like this one:

```
<owl:Ontology rdf:about=""/>
     <owl:Class rdf:ID="التعليمي المقرر">
      <rdfs:subClassOf>
         <owl:Class rdf:ID="نشاط"/>
      </rdfs:subClassOf>
    <rdfs:label xml:lang="ar">التعليمي المقرر</rdfs:label>
          <rdfs:comment xml:lang="en">education imparted in a series
of lessons or class meetings; "he took a course in basket weaving";
"flirting is not unknown in college classes"</rdfs:comment>
     </owl:Class>
     <owl:Class rdf:ID="CLASS_ROOT"/>
   <owl:Class rdf:ID="محتوى">
<rdfs:label xml:lang="ar">محتوى</rdfs:label>
      <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
      <rdfs:comment xml:lang="en">the sum or range of what has been
perceived, discovered, or learned</rdfs:comment>
   </owl:Class>
     <owl:Class rdf:ID="الانضباط">
          <rdfs:label xml:lang="ar">الانضباط</rdfs:label>
<rdfs:subClassOf rdf:resource="#محتوى"/>
      <rdfs:comment xml:lang="en">a branch of knowledge; "in what
discipline is his doctorate?"; "teachers should be well trained in
their subject"; "anthropology is the study of human
```

```
beings"</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="كلية ">
    <rdfs:comment xml:lang="en">the body of teachers and
administrators at a school; "the dean addressed the letter to the
entire staff of the university"</rdfs:comment>
      <rdfs:label xml:lang="ar">كلية </rdfs:label>
      <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
   </owl:Class>
   <owl:Class rdf:ID="برنامج">
<rdfs:subClassOf rdf:resource="#محتوى"/>
      <rdfs:comment xml:lang="en">a series of steps to be carried
out or goals to be accomplished; "they drew up a six-step plan";
"they discussed plans for a new bond issue"</rdfs:comment>
      <rdfs:label xml:lang="ar">برنامج</rdfs:label>
   </owl:Class>
     <owl:Class rdf:ID="عالم">
<rdfs:comment xml:lang="en">the concerns of the world as
distinguished from heaven and the afterlife; "they consider the
church to be independent of the world"</rdfs:comment>
          <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
          <rdfs:label xml:lang="ar">عالم</rdfs:label>
   </owl:Class>
     <owl:Class rdf:ID="برمجة">
      <rdfs:label xml:lang="ar">برمجة</rdfs:label>
      <rdfs:subClassOf>
         <owl:Class rdf:about="#نشاط"/>
      </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">setting an order and time for
planned events</rdfs:comment>
   </owl:Class>
     <owl:Class rdf:ID="لغة">
      <rdfs:label xml:lang="ar">لغة</rdfs:label>
      <rdfs:subClassOf>
         <owl:Class rdf:ID="إتصالات"/>
      </rdfs:subClassOf>
    <rdfs:comment xml:lang="en">the text of a popular song or
musical-comedy number; "his compositions always started with the
```

```
lyrics"; "he wrote both words and music"; "the song uses colloquial
language"</rdfs:comment>
     </owl:Class>
     <owl:Class rdf:ID="فـن">
          <rdfs:label xml:lang="ar">فـن</rdfs:label>
      <rdfs:comment xml:lang="en">photographs or other visual
representations in a printed publication; "the publisher was
responsible for all the artwork in the book"</rdfs:comment>
<rdfs:subClassOf>
               <owl:Class rdf:about="#إتـصـالات"/>
      </rdfs:subClassOf>
   </owl:Class>
     <owl:Class rdf:about="#إتـصـالات">
<rdfs:label xml:lang="ar">إتـصـالات</rdfs:label>
      <rdfs:subClassOf>
         <owl:Class rdf:ID="تـجريـد"/>
      </rdfs:subClassOf>
      <rdfs:comment xml:lang="en">something that is communicated by
or to or between people or groups</rdfs:comment>
     </owl:Class>
   <owl:Class rdf:about="#تـجريـد">
          <rdfs:label xml:lang="ar">تـجريـد</rdfs:label>
              <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
      <rdfs:comment xml:lang="en">a general concept formed by
extracting common features from specific examples</rdfs:comment>
     </owl:Class>
     <owl:Class rdf:about="#نـشـاط">
      <rdfs:subClassOf rdf:resource="#CLASS_ROOT"/>
      <rdfs:label xml:lang="ar">نـشـاط</rdfs:label>
          <rdfs:comment xml:lang="en">any specific activity; "they
avoided all recreational activity"</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="اسـتـخـدم">
    <rdfs:comment xml:lang="en">what something is used for; "the
function of an auger is to bore holes"; "ballet is beautiful but
what use is it?"</rdfs:comment>
      <rdfs:label xml:lang="ar">اسـتـخـدم</rdfs:label>
      <rdfs:subClassOf rdf:resource="#تـجريـد"/>
```

```
    </owl:Class>
    <owl:ObjectProperty rdf:ID="PROPERTY_HASA_ROOT"/>
    <owl:ObjectProperty rdf:ID="PROP_ROOT"/>
</rdf:RDF>
```

By using the Protégé editor, we can generate ontology graph for related Arabic words from specified text (Figure 4.6).

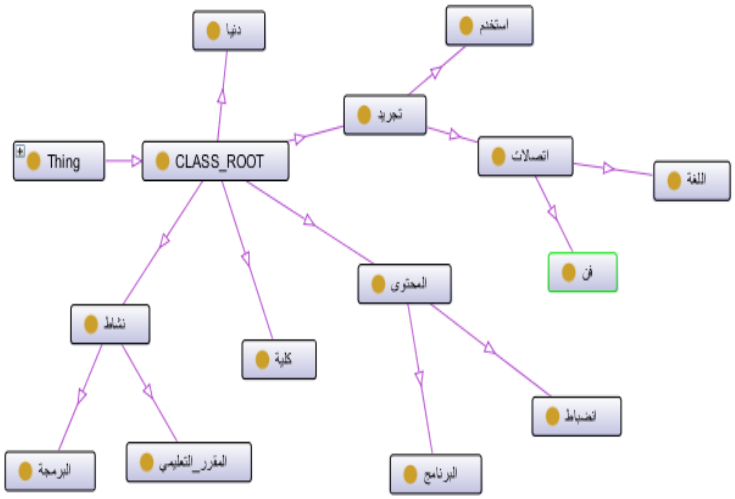

Figure 4.6 Ontology graph for Arabic words

The whole process is graphically presented in Figure 4.7.

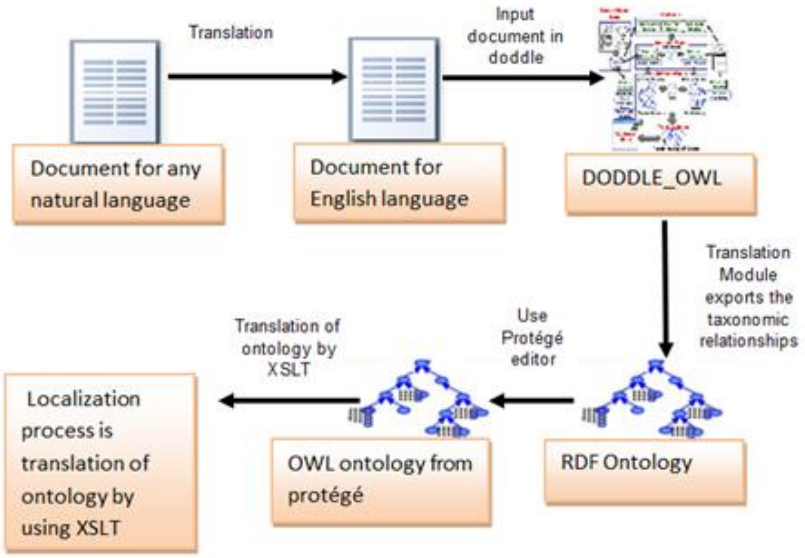

Figure 4.7 Building of NLO

4.6 Creating dictionary for the translation process

By applying DODDLE-OWL, after completing step 4, the project is completed and saved. When the project is saved, a file named "InputWordSet" is automatically created. This file contains all English words selected from Input Term Info Table.

This file consequently contains all relevant words for target ontology. These words should be translated into corresponding words in target language. Here we utilize MyMemory online translation service [69]. MyMemory is the world's largest Translation Memory. It has been created collecting TMs from the European Union, United Nations and aligning the best domain specific multilingual websites.

MyMemory's translation service is accessible over the Internet via their translation API. We wrote WordTranslator console application in .NET Framework 4.5 which utilizes their translation API. Performing translation of a single word is done by calling WordTranslator with three following arguments: word to be translated, language of the provided word and the desired language for the translation. Program outputs a single translated word. This greatly simplifies translation process, since complexities of natural language translation are reduced to a single console command execution. (If some words are not translated, then we use Google translator or any available tool for translation.)

WordTranslator can be used for any combination of input/output languages. Since we are starting with English ontologies, all our input words will be in English, and output in the target language. We use outputs generated by WordTranslator to construct dictionary used by XSLT transformation. Format of the dictionary is following one (mapping English to Arabic words):

```
<?xml version="1.0" encoding="UTF-8"?>
<dictionary>
  <word name="world">عالم </word><word name="program">برنامج</word>
  <word name="course">صنف</word><word name="use">استخدم</word>
  <word name="discipline">النظام</word>
  <word name="art">فن</word><word name="programming">برمجة</word>
    ……
  </dictionary>
```

After building English OWL representation of our text in step 5, we transform this document into Arabic OWL representation by using XSLT transformer. For this

purpose we use a XML editor. There are more available XML editors (see: [70] and [71]). For example, by using Oxygen XML Editor [71], we get the workspace organized as in Figure 4.8.

Figure 4.8 XSLT transformer applied by Oxygen XML Editor

4.7 Examples

In this section we present two examples applying our procedure for Serbian and French. We started from the following Serbian text:

Програмирање је лепа уметност. Програмирање је курс на Математичком факултету. Програмирање је веома ефикасна дисциплина и учи се на многим факултетима у свету. Сада можемо да користимо много нових програмских језика и направити различите програме.

A part of XSLT transformation for this text has the form as in Figure 4.9.

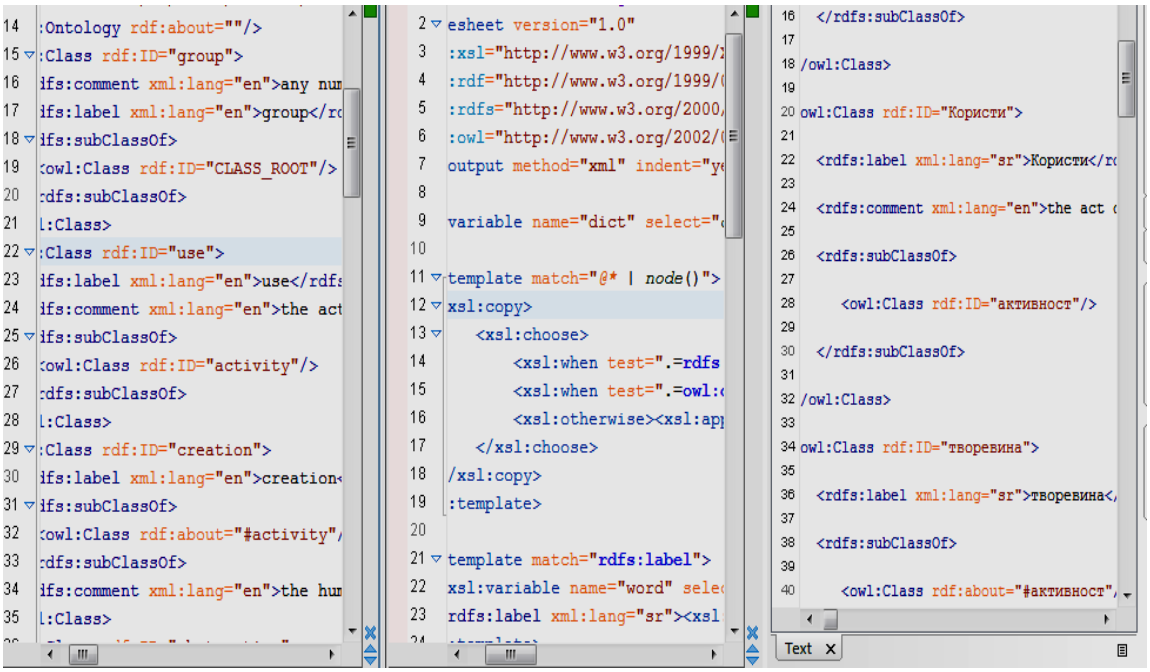

Figure 4.9 XSLT transformer applied for Serbian text

A piece of obtained ontology for Serbian text from above (the entire document is too long) looks as following:

```
<owl:Ontology rdf:about=""/>
   <owl:Class rdf:ID="курс">
      <rdfs:subClassOf>
         <owl:Class rdf:ID="активност"/>
      </rdfs:subClassOf>
      <rdfs:label xml:lang="sr">курс</rdfs:label>
      <rdfs:comment xml:lang="en">education imparted in a series of
lessons or class meetings; "he took a course in basket weaving";
"flirting is not unknown in college classes"</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="КЛАСА_РООТ"/>
   <owl:Class rdf:ID="садржај">
      <rdfs:label xml:lang="sr">садржај</rdfs:label>
      <rdfs:subClassOf rdf:resource="#КЛАСА_РООТ"/>
      <rdfs:comment xml:lang="en">the sum or range of what has been
perceived, discovered, or learned</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="дисциплина">
      <rdfs:label xml:lang="sr">дисциплина</rdfs:label>
```

```
      <rdfs:subClassOf rdf:resource="#садржај"/>
      <rdfs:comment xml:lang="en">a branch of knowledge; "in what
discipline is his doctorate?"; "teachers should be well trained in
their subject"; "anthropology is the study of human
beings"</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="факултет">
      <rdfs:comment xml:lang="en">the body of teachers and
administrators at a school; "the dean addressed the letter to the
entire staff of the university"</rdfs:comment>
      <rdfs:label xml:lang="sr">факултет</rdfs:label>
      <rdfs:subClassOf rdf:resource="#КЛАСА_РООТ"/>
   </owl:Class>
   <owl:Class rdf:ID="програм">
      <rdfs:subClassOf rdf:resource="#садржај"/>
      <rdfs:comment xml:lang="en">a series of steps to be carried
out or goals to be accomplished; "they drew up a six-step plan";
"they discussed plans for a new bond issue"</rdfs:comment>
      <rdfs:label xml:lang="sr">програм</rdfs:label>
   </owl:Class>
   <owl:Class rdf:ID="текст">
       <rdfs:comment xml:lang="en">the concerns of the world as
distinguished from heaven and the afterlife; "they consider the
church to be independent of the world"</rdfs:comment>
      <rdfs:subClassOf rdf:resource="#КЛАСА_РООТ"/>
      <rdfs:label xml:lang="sr">текст</rdfs:label>
   </owl:Class>
   <owl:Class rdf:ID="програмирање">
      <rdfs:label xml:lang="sr">програмирање</rdfs:label>
      <rdfs:subClassOf>
         <owl:Class rdf:about="#активност"/>
      </rdfs:subClassOf>
      <rdfs:comment xml:lang="en">setting an order and time for
planned events</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="језик">
      <rdfs:label xml:lang="sr">језик</rdfs:label>
      <rdfs:subClassOf>
```

```
         <owl:Class rdf:ID="комуникација"/>
      </rdfs:subClassOf>
      <rdfs:comment xml:lang="en">the text of a popular song or
musical-comedy number; "his compositions always started with the
lyrics"; "he wrote both words and music"; "the song uses colloquial
language"</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="уметност">
      <rdfs:label xml:lang="sr">уметност</rdfs:label>
      <rdfs:comment xml:lang="en">photographs or other visual
representations in a printed publication; "the publisher was
responsible for all the artwork in the book"</rdfs:comment>
       <rdfs:subClassOf>
         <owl:Class rdf:about="#комуникација"/>
      </rdfs:subClassOf>
   </owl:Class>
   <owl:Class rdf:about="#комуникација">
<rdfs:label xml:lang="sr">комуникација</rdfs:label>
      <rdfs:subClassOf>
         <owl:Class rdf:ID="апстракција"/>
      </rdfs:subClassOf>
      <rdfs:comment xml:lang="en">something that is communicated by
or to or between people or groups</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:about="#апстракција">
      <rdfs:label xml:lang="sr">апстракција</rdfs:label>
 <rdfs:subClassOf rdf:resource="#КЛАСА_ROOT"/>
      <rdfs:comment xml:lang="en">a general concept formed by
extracting common features from specific examples</rdfs:comment>
   </owl:Class>
     <owl:Class rdf:about="#активност">
      <rdfs:subClassOf rdf:resource="#КЛАСА_ROOT"/>
      <rdfs:label xml:lang="sr">активност</rdfs:label>
        <rdfs:comment xml:lang="en">any specific activity; "they
avoided all recreational activity"</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="користити">
<rdfs:comment xml:lang="en">what something is used for; "the
```

```
function of an auger is to bore holes"; "ballet is beautiful but
what use is it?"</rdfs:comment>
      <rdfs:label xml:lang="sr">користити</rdfs:label>
      <rdfs:subClassOf rdf:resource="#апстракција"/>
   </owl:Class>
   <owl:ObjectProperty rdf:ID="PROPERTY_HASA_ROOT"/>
   <owl:ObjectProperty rdf:ID="PROP_ROOT"/>
</rdf:RDF>
```

By using the Protégé editor, we can generate ontology graph for related Serbian word from specified text (Figure 4.10). From this graph we can see relations between concepts in given text.

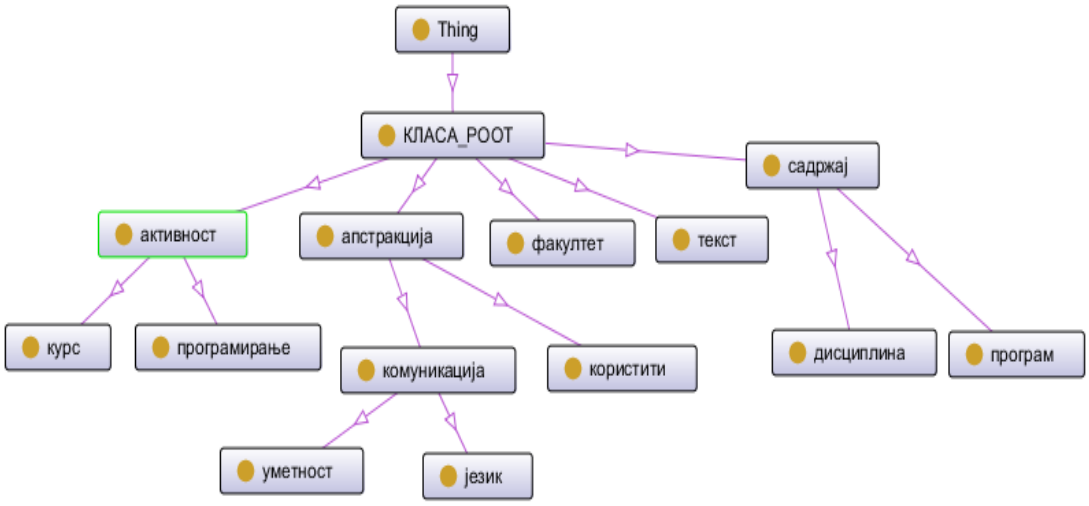

Figure 4.10 Ontology graph for Serbian words

Example of using the same content (text), with French the target language:

La programmation est un art magnifique. La programmation est un cours à la Faculté de Mathématiques. La programmation est une discipline très efficace et il est appris dans de nombreux collèges dans le monde. Maintenant, nous pouvons utiliser un grand nombre de nouveaux langages de programmation et de faire beaucoup de différents programmes.

Part of XSLT transformation is shown in Figure 4.11.

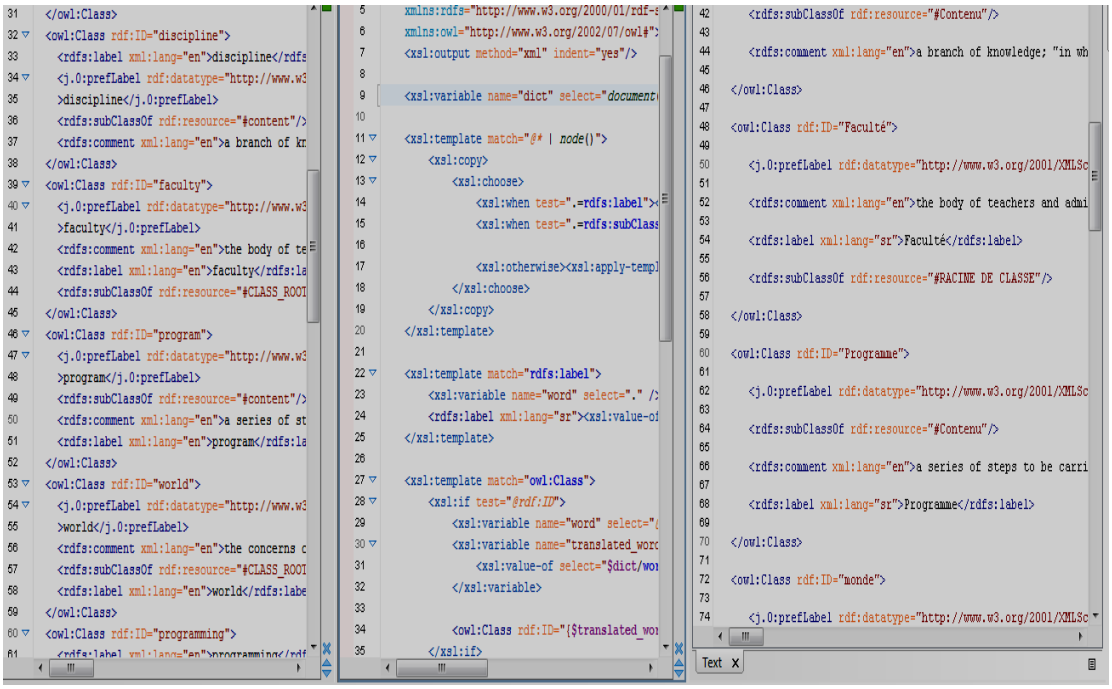

Figure 4.11 XSLT transformer applied for French language

A fraction of OWL document for French language looks like this:

```
<owl:Ontology rdf:about=""/>
     <owl:Class rdf:ID="cours">
      <rdfs:subClassOf>
         <owl:Class rdf:ID="Activité"/>
      </rdfs:subClassOf>
    <rdfs:label xml:lang="fr">cours</rdfs:label>
      <rdfs:comment xml:lang="en">education imparted in a series of
lessons or class meetings; "he took a course in basket weaving";
"flirting is not unknown in college classes"</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="RACINE DE CLASSE"/>
   <owl:Class rdf:ID="Contenu">
 <rdfs:label xml:lang="fr">Contenu</rdfs:label>
      <rdfs:subClassOf rdf:resource="#RACINE DE CLASSE"/>
      <rdfs:comment xml:lang="en">the sum or range of what has been
perceived, discovered, or learned</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="Discipline">
      <rdfs:label xml:lang="fr">Discipline</rdfs:label>
<rdfs:subClassOf rdf:resource="#Contenu"/>
```

```
      <rdfs:comment xml:lang="en">a branch of knowledge; "in what
discipline is his doctorate?"; "teachers should be well trained in
their subject"; "anthropology is the study of human
beings"</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="Faculté">
        <rdfs:comment xml:lang="en">the body of teachers and
administrators at a school; "the dean addressed the letter to the
entire staff of the university"</rdfs:comment>
      <rdfs:label xml:lang="fr">Faculté</rdfs:label>
      <rdfs:subClassOf rdf:resource="#RACINE DE CLASSE"/>
   </owl:Class>
   <owl:Class rdf:ID="Programme">
      <rdfs:subClassOf rdf:resource="#Contenu"/>
      <rdfs:comment xml:lang="en">a series of steps to be carried
out or goals to be accomplished; "they drew up a six-step plan";
"they discussed plans for a new bond issue"</rdfs:comment>
      <rdfs:label xml:lang="fr">Programme</rdfs:label>
   </owl:Class>
   <owl:Class rdf:ID="monde">
<rdfs:comment xml:lang="en">the concerns of the world as
distinguished from heaven and the afterlife; "they consider the
church to be independent of the world"</rdfs:comment>
      <rdfs:subClassOf rdf:resource="#RACINE DE CLASSE"/>
      <rdfs:label xml:lang="fr">monde</rdfs:label>
   </owl:Class>
   <owl:Class rdf:ID="programmation">
      <rdfs:label xml:lang="fr">programmation</rdfs:label>
      <rdfs:subClassOf>
         <owl:Class rdf:about="#Activité"/>
      </rdfs:subClassOf>
<rdfs:comment xml:lang="en">setting an order and time for planned
events</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="langage">
      <rdfs:label xml:lang="fr">langage</rdfs:label>
      <rdfs:subClassOf>
         <owl:Class rdf:ID="Communication."/>
```

```
      </rdfs:subClassOf>
          <rdfs:comment xml:lang="en">the text of a popular song or
musical-comedy number; "his compositions always started with the
lyrics"; "he wrote both words and music"; "the song uses colloquial
language"</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="l'art.">
      <rdfs:label xml:lang="fr">l'art.</rdfs:label>
      <rdfs:comment xml:lang="en">photographs or other visual
representations in a printed publication; "the publisher was
responsible for all the artwork in the book"</rdfs:comment>
<rdfs:subClassOf>
         <owl:Class rdf:about="#Communication."/>
      </rdfs:subClassOf>
   </owl:Class>
   <owl:Class rdf:about="#Communication.">
      <rdfs:label xml:lang="fr">Communication.</rdfs:label>
      <rdfs:subClassOf>
         <owl:Class rdf:ID="abstraction"/>
      </rdfs:subClassOf>
      <rdfs:comment xml:lang="en">something that is communicated by
or to or between people or groups</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:about="#abstraction">
      <rdfs:label xml:lang="fr">abstraction</rdfs:label>
      <rdfs:subClassOf rdf:resource="#RACINE DE CLASSE"/>
      <rdfs:comment xml:lang="en">a general concept formed by
extracting common features from specific examples</rdfs:comment>
   </owl:Class>
     <owl:Class rdf:about="#Activité">
      <rdfs:subClassOf rdf:resource="#RACINE DE CLASSE"/>
      <rdfs:label xml:lang="fr">Activité</rdfs:label>
 <rdfs:comment xml:lang="en">any specific activity; "they avoided
all recreational activity"</rdfs:comment>
   </owl:Class>
   <owl:Class rdf:ID="utiliser">
<rdfs:comment xml:lang="en">what something is used for; "the
function of an auger is to bore holes"; "ballet is beautiful but
```

```
what use is it?"</rdfs:comment>
      <rdfs:label xml:lang="fr">utiliser</rdfs:label>
      <rdfs:subClassOf rdf:resource="#abstraction"/>
   </owl:Class>
   <owl:ObjectProperty rdf:ID="PROPERTY_HASA_ROOT"/>
   <owl:ObjectProperty rdf:ID="PROP_ROOT"/>
</rdf:RDF>
```

We applied Protégé editor, as in previous example to get ontology graph for French text (Figure 4.12).

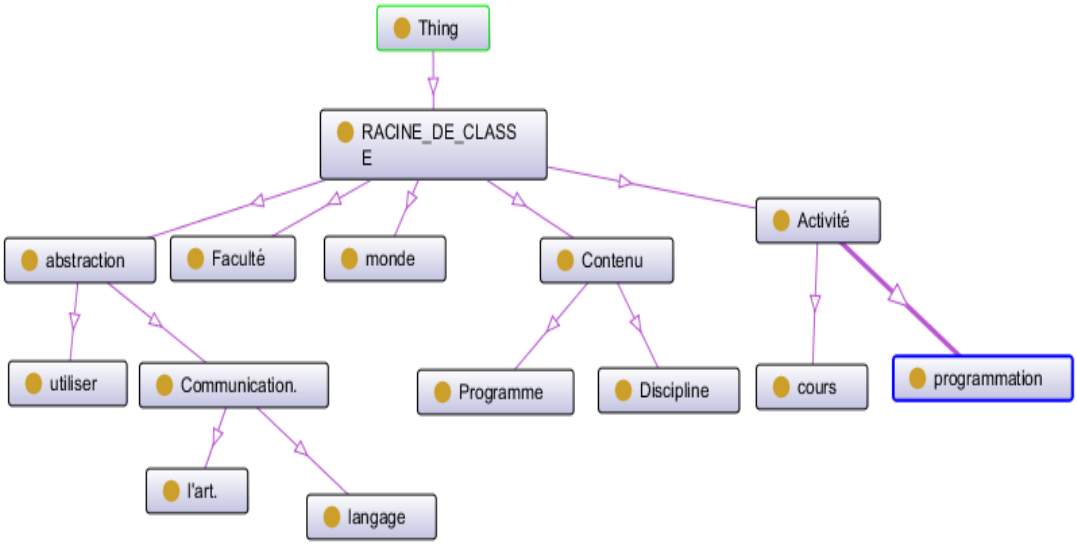

Figure 4.12 Ontology graph for French words

Consider another more complex example is presented in Fig. 13. Let us have the following French text:

L'informatique est le domaine d'activité scientifique, technique et industriel concernant le traitement automatique de l'information via l'exécution de programmes informatiques par des machines: des systèmes embarqués, des ordinateurs, des robots, des automates, etc.

Ces champs d'application peuvent être séparés en deux branches, l'une, de nature théorique, qui concerne la définition de concepts et modèles, et l'autre, de nature pratique, qui s'intéresse aux techniques concrètes d'implantation et de mise en œuvre sur le terrain. Certains domaines de l'informatique peuvent être très abstraits, comme la complexité algorithmique, et d'autres peuvent être plus proches d'un public

profane. Ainsi, la théorie des langages demeure un domaine davantage accessible aux professionnels formés (description des ordinateurs et méthodes de programmation), tandis que les métiers liés aux interfaces homme-machine sont accessibles à un plus large public.

Le terme « informatique » résulte de la combinaison des trois premières syllabes du terme « information » et des deux dernières syllabes du terme « automatique » ; il désigne à l'origine l'ensemble des activités liées à la conception et à l'emploi des ordinateurs pour traiter des informations. Dans le vocabulaire universitaire américain, il désigne surtout l'informatique théorique : un ensemble de sciences formelles qui ont pour objet d'étude la notion d'information et des procédés de traitement automatique de celle-ci, l'algorithmique. Par extension, la mise en application de méthodes informatiques peut concerner des problématiques annexes telles que le traitement du signal, la calculabilité ou la théorie de l'information.

After applying our method, the following ontology graph is obtained (we omit intermediate ontology graph for English words and parts of XSLT transformer):

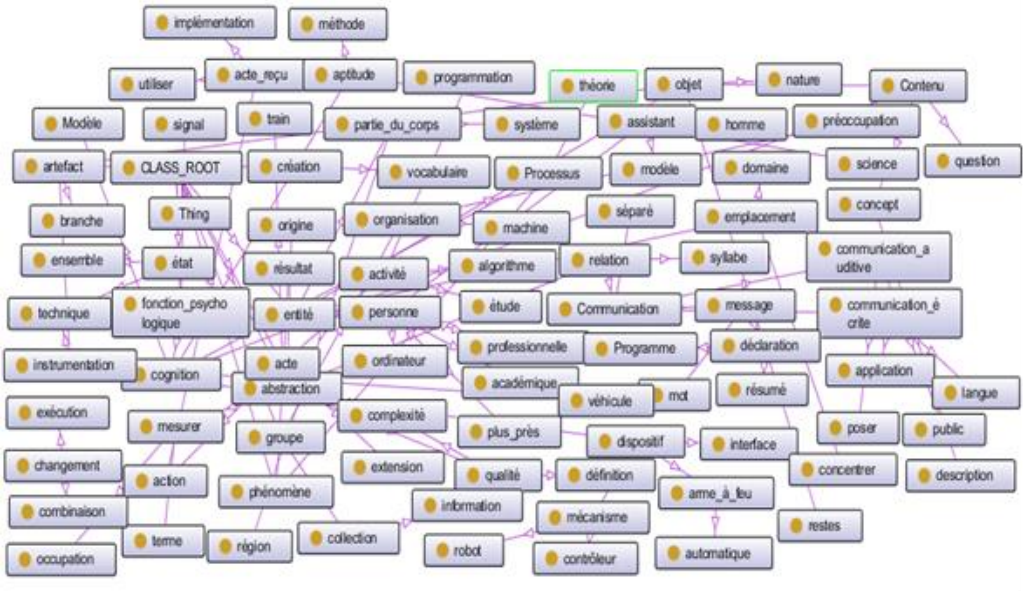

Figure 4.13 Ontology graph for more complex French words

Conclusion

Ontologies are very important in different scientific fields such as: knowledge engineering and representation, information retrieval and extraction, knowledge management, agent systems, and so on. We can say that ontologies represent the backbone of the semantic web. The possibility to create ontology for any natural language gives us an opportunity to work with information that can be processed by both humans and computers in a natural way. This is, unfortunately, still difficult to do automatically. However, semi-automatic implementation of this process, including a human expert, is possible. We described our approach for discovering taxonomic conceptual relations from text facility ontology by using open source software tool DODDLE-OWL. The main challenge we faced is that this software is available only for Japanese and English languages. To address that, we proposed the procedure where DODDLE-OWL is used as an auxiliary tool to build an ontology from the given text for any natural language. For this approach the other auxiliary tools are necessary as well as an existing WordNet database for the target language, Protégé semantic web editor and Oxygen XML Editor. The main contribution of this chapter is the integration of different software tools, which gives new quality and provides the building of ontologies for different natural languages. We plan to perform further analysis of the results and compare the obtained ontology trees using different natural languages with the same input text. We will try to improve the proposed approach by integrating additional software tools and making certain steps simpler.

Chapter 5

Conclusion

We are mainly concerned with presenting the basic concepts of ontology and semantic web. Here is a summary of the main points discussed:

- The semantic web is established as layers and ontology is the core element of the Semantic Web. Ontology is an explicit and formal specification of a shared conceptualization. Ontologies can be classified according to their level of formalization. Designing an ontology is an iterative process. There are several ontology specification languages with different characteristics among them: XML, DAML + OIL, RDF, OWL and F-Logic.
- The Apollo, Protégé 3.4 and Swoop tools are open source ontology. Examples of these tools are: OntoStudio and TopBraid composer. A (FE) software license is requested. Application ontology demands learning / knowledge of a particular language Swoop. The tools: Protégé, TopBraid Composer (FE) and OntoStudio use databases for storing ontologies. The same applications are used to backup management functionality, which is just provided by TopBraid Composer (FE).

  Protégé and OntoStudio are also considered as graphical ontology tools. The Swoop is Web-based application. OntoStudio gives support to the Onto Knowledge methodology. TopBraid Composer (FE) uses the Exception Handling. Some of the tools only support the joint edition of the functions of browsing. Protégé, TopBraid Composer (FE), Swoop and OntoStudio editors provide documentation ontology, ontology import / export to different formats, graphical view of ontologies, ontology libraries and attached inference engines and Apollo supports Apollo metalanguage.

  - Being the backbone of the semantic web, ontologies are of many important uses. The possibility to create ontology for any natural language gives an opportunity to work with pieces of information that are accessible for both humans and computers. Unfortunately, it is still

difficult to do it automatically. Moreover, semi-automatic implementation of this process (where human expert is included) is possible. The approach for discovering taxonomic conceptual relations from text facility ontology is described by using open source software tool DODDL-OWL.

The main problem lies in the fact that this software is available only for Japanese and English languages. Therefore, the researcher proposes the procedure where DODDL-OWL is used as auxiliary tool to build the ontology from given text for any natural language (target language). Some other auxiliary tools are necessary too. For example, WordNet database for the target language, Protégé semantic web editor and Oxygen XML Editor. The main contribution of chapter (4) is the integration of different software tools, which gives new quality and provides the building of ontologies for different natural languages. We plan to perform further analysis of the results and compare the obtained ontology trees using different natural languages with the same input text. We will try to improve the proposed approach by integrating additional software tools and making certain steps simpler.

References

[1] T. R. Gruber and T. R. Gruber, "A Translation Approach to Portable Ontology Specifications by A Translation Approach to Portable Ontology Specifications," vol. 5, no. April, pp. 199-220, 1993.

[2] Vesin, B., Ivanović, M., Klašnja-Milićević, A., Budimac, Z.: Ontology-Based Architecture with Recommendation Strategy in Java Tutoring System. Computer Science and Information Systems, Vol. 10, No. 1, 237-261. (2013)

[3] Wei, M., Xu, J., Yun, H., Xu, L.: Ontology-Based Home Service Model. Computer Science and Information Systems, Vol. 9, No. 2, 813-838. (2012)

[4] Tim Berners-Lee, James Hendler and Ora Lassila: The Semantic Web, Scientific American, 2001

[5] Berners-Lee, T. (2001). The semantic web. *Scientific american* , pp. 284(5):35–43.

[6] J. Cai and V. Eske, "1 . Semantic Web: Informational Retrieval System," pp. 1–30.

[7] I. J. Doyle, P. Torasso, E. Sandewall, F. International, K. Representation, G. S. Institut, T. R. Gruber, and G. R. Olsen, "An Ontology for Engineering Mathematics 2 . The Purpose of the Ontology," 1994.

[8] N. Guarino and M. Carrara, "An Ontology of Meta-Level Categories.".

[9] G. Guizzardi, "Modeling Languages, and (Meta) Models.".

[10] M. Ferndndez, A. Gmez-p, and N. Juristo, "METHONTOLOGY: From Ontological Art Towards Ontological Engineering," pp. 33–40, 1997.

[11] R. Mizoguchi and M. Ikeda, "Towards Ontology Engineering," pp. 1–10.

[12] Studer, R., Benjamins, R., Fensel, D. Knowledge Engineering: Principles and Methods.DKE 25(1-2).pp:161-197. 1998

[13] Oscar Corcho and Asuncion Gomez-Perez, A Roadmap to Ontology Specification Languages, in Rose Dieng and Olivier Corby (eds.), Knowledge Engineering and Knowledge Management. Methods, Models and Tools, Springer, Berlin, 80-96, 2000.

[14] Components of ontology.http://ontogenesis.knowledgeblog.org/514?kblog-transclude=2#_concept.

[15] Gruber, R. A translation approach to portable ontology specification . KnowledgeAcquisition. #5: 199-220. 1993.

[16] Motta, E. Reusable Components for Knowledge Modelling. IOS Press. Amsterdam. 1999.

[17] Farquhar, A., Fikes, R., Rice, J. The Ontolingua Server: A Tool for Collaborative Ontology Construction. Proceedings of KAW96. Banff, Canada, 1996

[18] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann, and M. Klein, "OIL in a Nutshell."].

[19] J. P. Tim Bray and E. M. and F. Y. C.M Sperberg-McQueen, "Extensible Markup Language (XML) 1.0," *W3C ...*, 2008. Online. Available: http://www.w3.org/TR/REC-xml/. [Accessed: 26-Dec-2012].

[20] J. Z. Pan, "DESCRIPTION LOGICS: REASONING SUPPORT FOR," 2004. 6],

[21] D. L. M. Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, B. L. R. Peter F. Patel-Schneider, and L. A. Stein, "OWL Web Ontology Language Reference," 2004. [Online]. Available: http://www.w3.org/TR/owl-ref/.

[22] D. L. M. Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, B. L. R. Peter F. Patel-Schneider, and L. A. Stein, "OWL Web Ontology Language Reference," 2004. [Online]. Available: http://www.w3.org/TR/owl-ref/.

[23] M. Kifer, G. Lausen, and J. Wu, "Logical foundations of object-oriented and frame-based languages," *Journal of the ACM*, vol. 42, no. 4, pp. 741–843, Jul. 1995.

[24] Akkemans, M., and Yagge, F. (1999) 'Decentralized Markets versus Central Control: A Comparative Study', *Journal of Artificial Inteligence*, Vol. 11, pp.301-33.

[25] Maedche, A.and Stabb, S. (2001) 'Knowledge Portals— Ontologies at Work', *AI Magazine*, Vol. 21, no.2

[26] Grimm, S., Hitzler, P., And Abecker, A. (2007). Knowledge Representation and Ontologies Logic, Ontologies and Semantic Web Languages.Springer-Verlag, ch. 3, pp. 51–106.

[27] Guarino, N. Semantic Matching: Formal ontological distinctions for information organization, extraction and integration. In information extraction: A multidisciplinary approach to an emerging information technology (1997), M.T. Pazienza, Ed., Springer Verlag, pp. 139-170.

[28] Guarino, N. Formal ontology and information systems. In proceeding of the 1st international conference on formal ontologies in information systems(FOIS'89) (Trento, Italy,1998),IOS Press, PP. 3-15.

[29] Stamatios A. Theocharis, George A. Tsihrintzis, "Semantic Web Technologies in e - Government,". pp. 1237–1244, 2012.

[30] Hendrix, G.G., & Carbonell, J.G. (1981). A Tutorial on Natural Language Processing. *ACM. '81*, 4-8.

[31] Alani, H., Hara, K.O., Shadbolt, N., "Common features of killer apps: A comparison with Protégé", In: 8th International Prot Conference, 18-21 July 2005, Madrid, Spain.

[32]Florida, W., Mulholland, P., Tyler, N., Hoffman, R., Genest, D., "Table 1. Ontology editor survey results", from: http://www.xml.com/2004/07/14/examples/Ontology_Editor_Survey_2004_Table_-_Michael_Denny.pdf.

[33] Karim, S., Tjoa, A.M., "Towards the Use of Ontologies for Improving User Interaction for People with Special Needs", in: Computers Helping People with Special Needs, vol. 4061/2006, (pp. 77–84). Berlin: Springer-Heidelberg.

[34] Lau, T. and Sure, Y. (2002) Introducing ontology-based skills management at a large insurance company. In Proceedings of the Modellierung 2002, March, Tutzing, Germany.

[35] Buraga, S.C., Cojocaru, L., Nichifor, O.C., "Survey on Web Ontology Editing Tools", Science,

[36] Kapoor, B., Sharma, S., "A Comparative Study Ontology Building Tools for Semantic Web Applications", International Journal 1, July (2010), 1-13.

[37] Funk, A., Tablan, V., Bontcheva, K., Cunningham, H., Davis, B., Handschuh, S., "CLOnE: Controlled Language for Ontology Editing", Most, (2007), 142-155.

[38] Stojanovic L., Motik B., "Ontology Evolution within Ontology Editors", from: http://km.aifb.kit.edu/ws/eon2002/EON2002_Stojanovic.pdf

[39] Suresh K., Malik S. K., Prakash N. Rizvi S., "Role of Ontology Editors: Ontology Design", from: http://nguyendangbinh.org/Proceedings/IPCV08/Papers/SWW4578.pdf

[40] Kashyap, V., Bussler, C., Matthew, M., "The Semantic Web: Semantics for Data and Services on the Web (Data-Centric Systems and Applications)", (pp-136), Springer-Verlag Berlin Heidelberg

[41] Ontology building editor developed by Knowledge Media Institute, The Open University, further information available, from: http://apollo.open.ac.uk/.

[42] Ontostudio, W. & Information, E. OntoStudio 3 Professional tool for knowledge architects, from:http://www.ontoprise.de/fileadmin/user_upload/Flyer_EN/ Flyer_Onto Studio_en.pdf .

[43] Weiten, M., "OntoSTUDIO as Ontology Engineering Ontology Engineering Tools: State of the Art", Journal of Web Semantics, 51-60. (Weiten, M)

[44] Protégé ontology editor developed by Stanford Medical Informatics, Stanford University School of Medicine, further information available, from: http://protege.stanford.edu/.

[45] Knublauch, H., Fergerson, R.W., Noy, N.F., Musen, M.A. "The protégé OWL Plugin: An Open Development Environment for Semantic Web Applications", Third International Semantic Web Conference (ISWC 2004), Germany. Berlin: Springer-Heidelberg.

[46] Cardoso, J., "Editing Tools for Ontology Construction", Idea, March (2007), 1-27.

[47] Collaborative protégé, from: http://protegewiki.stanford.edu/wiki/Collaborative_Protege.

[48] Swoop. (2004) "Semantic Web Ontology Overview and Perusal", from:

http://www.mindswap.org/2004/SWOOP/.

[49] Kalyanpur, A., Parsia, B., Sirin, E., Grau, B.C., Hendler, J., “Swoop: A ‘Web’ Ontology Editing Browser”, Mind, July 2005, 1-20.

[50] Top Braid Composer, from: http://www.topquadrant.com/products/TB_Composer.html

[51] OWL 2 Support in TopBraid Composer, from: http://composing-the-semantic-web.blogspot.com/2009/10/owl-2-support-in-topbraid-composer.html.

[52] TopBraid, from: http://www.w3.org/2001/sw/wiki/TopBraid.

[53] P. Cimiano, J. Völker, "Text2Onto - A Framework for Ontology Learning and Data-driven Change Discovery".In Andres Montoyo, Rafael Munoz, Elisabeth Metais, *Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB)*, pages: 227- 238, Springer, Lecture Notes in Computer Science, 3513, Alicante, Spain, Juni, 2005.

[54] S. Mittal , N. Mittal, " Tools for Ontology Building from Texts: Analysis and Improvement of the Results of Text2Onto". IOSR Journal of Computer Engineering (IOSR-JCE),e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 11, Issue 2 (May. - Jun. 2013), PP 101-117.

[55] N. Shadbolt, K. O'hara and L. Crow," The experimental evaluation of knowledge acquisition techniques and methods" , history , problems and new directions, 729–755. (1999).

[56] D. Faure, and C. Nedellec, "Knowledge acquisition of predicate argument structures from technical texts using Machine Learning", the system asium, 2–7. (1999).

[57] A.G. Nathalie, S. Despres.and S. Szulman. "The TERMINAE Method and Platform for Ontology Engineering from Texts".Proceedings of the 2008 conference on Ontology Learning and Population: Bridging the Gap between Text and Knowledge Pages 199-223.

[58] DODDLE-OWL,a Domain Ontology rapiD DeveLopment Environment - OWL extension, [Online]. Available: http://doddle-owl.sourceforge.net/en/.

[59] Morita, T., Izumi, N., Fukuta, N.: Special Section on Knowledge-Based ++Tool. doi:10.1093/ietisy/e89.

[60] Takeshi, M., Naoki, F., Noriaki I., Takahira, Y.: DODDLE-OWL: Interactive Domain Ontology Development with Open Source Software in Java. IEICE Transactions 91-D(4): 945-958. (2008)

[61] Takeshi, M., Naoki, F., Noriaki I., Takahira, Y.: DODDLE-OWL: A Domain Ontology Construction Tool with OWL. ASWC 2006: 537-551.(2006)

[62] C.Fellbaum, "WordNet", The MIT Press, 1998. See also URL: http://www.cogsci.princeton.edu/~wn/ .

[63] Gruber T.: *Towards principles for the design of ontologies used for knowledge sharing*. Int. Journal of Human and Computer Studies 43(5/6), 1994, 907-928.

[64] Maedche A.: *Ontology Learning for the Semantic Web*. The Kluwer International Series in Engineering and Computer Science, Volume 665, 2003.

[65] Gomez-Perez A. / Manzano-Macho D.: *A Survey of Ontology Learning Methods and Techniques*. Deliverable 1.5, OntoWeb Project, 2003.

[66] Hamish Cunningham, Diana Maynard, K. B. and V. T. (2002). GATE : A framework and graphical development environment for robust NLP tools and applications

[67] Takeshi, M., Noriaki, I., Naoki F., Takahira, Y.: A Graphical RDF-Based Meta-Model Management Tool. IEICE Transactions 89-D(4): 1368-1377. (2006)

[68] F.Z. Belkredim and A. E. Sebai, "An Ontology Based Formalism for the Arabic Language Using Verbs and their Derivatives". 11, 44–52. (2009).

[69] MyMemory: next generation Translation Memory technology, [Online]. Available: http://mymemory.translated.net/doc/.

[70] Microsoft Core XML Services (MSXML) 6.0, [Online]. Available: http://www.microsoft.com/enus/download/details.aspx?id=3988.

[71] Oxygen XML Editor 14.2. [Online]. Available: http://www.oxygenxml.com/doc/Editor-UserManual.pdf

Прилог 1.

Изјава о ауторству

Потписани-а _ Emhimed alatrish

број уписа _____2033/2009

Изјављујем

да је докторска дисертација под насловом

“**Using Web Tools for Constructing an Ontology of Different Natural Languages**”

- резултат сопственог истраживачког рада,
- да предложена дисертација у целини ни у деловима није била предложена за добијање било које дипломе према студијским програмима других високошколских установа,
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио интелектуалну својину других лица.

Потпис докторанда

У Београду, _____

_____

Прилог 2.

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора __Emhimed alatrish

Број уписа _____2033/2009_____

Студијски програм computer science _____

Наслов рада "**Using Web Tools for Constructing an Ontology of Different Natural Languages**"

Ментор Prof. Dr. Dušan D. Tošić

Потписани __Emhimed alatrish

изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла за објављивање на порталу **Дигиталног репозиторијума Универзитета у Београду**.

Дозвољавам да се објаве моји лични подаци везани за добијање академског звања доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис докторанда

У Београду, _____

_____

Прилог 3.

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

“Using Web Tools for Constructing an Ontology of Different Natural Languages”

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигитални репозиторијум Универзитета у Београду могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство

2. Ауторство - некомерцијално

3. Ауторство – некомерцијално – без прераде

4. Ауторство – некомерцијално – делити под истим условима

5. Ауторство – без прераде

6. Ауторство – делити под истим условима

(Молимо да заокружите само једну од шест понуђених лиценци, кратак опис лиценци дат је на полеђини листа).

Потпис докторанда

У Београду, _____

_____