Semantic Web Tools and Techniques for Knowledge Organization: An Overview†

Thimmaiah Padmavathi* and Madaiah Krishnamurthy**

*CSIR-Central Food Technological Research Institute (CSIR-CFTRI), Food Science & Technology Information Services (FOSTIS/Library), Mysore, Karnataka, India, <padmavathit@yahoo.com>
**Documentation Research & Training Centre (DRTC), Indian Statistical Institute, Bangalore, Karnataka, <mkrishna_murthy@hotmail.com>

ThimmaiahPadmavathi is Senior Technical Officer at CSIR-Central Food Technological Research Institute, FOSTIS/Library. She holds BSc and MLISc from Bangalore University, completed a post master's training course in information technology applications to library and information services at the National Center for Science Information, Indian Institute of Science, Bangalore, and a doctoral degree from Bharathiar University. Her core areas of expertise include semantic web technologies, computer based information services, digital library technologies, documentation, IT applications (web page design and content development) and library automation.

Madaiah Krishnamurthy is an associate professor of information science at DRTC, Indian Statistical Institute, Bangalore. He holds a master's degree in applied economics and library and information science and a PhD from Bangalore. He was a Fulbright Scholar in 2006 and visited the Graduate School of Library and Information Science, University of Illinois at Urbana-Champaign (USA), and was recipient of ETD travel grant to attend the 16th International Symposium on Electronic Theses and Dissertations in France in 2016. He is author of over seventy articles. His research interests are digital libraries, institutional repositories, social networking, library management and automation.

Padmavathi, Thimmaiah and Madaiah Krishnamurthy. 2017. "Semantic Web Tools and Techniques for Knowledge Organization: An Overview." *Knowledge Organization* 44(4): 273-290. 46 references.

Abstract: The enormous amount of information generated every day and spread across the web is diversified in nature far beyond human consumption. To overcome this difficulty, the transformation of current unstructured information into a structured form called a "Semantic Web" was proposed by Tim Berners-Lee in 1989 to enable computers to understand and interpret the information they store. The aim of the semantic web is the integration of heterogeneous and distributed data spread across the web for knowledge discovery. The core of sematic web technologies includes knowledge representation languages RDF and OWL, ontology editors and reasoning tools, and ontology query languages such as SPARQL have also been discussed.

† The authors wish to thank the director, CSIR-CFTRI for his support and encouragement.

Received: 28 November 2016; Revised: 12 March 2017; Accepted 17 May 2017

Keywords: ontologies, OWL, query, domain knowledge, semantic web, knowledge discovery, knowledge representation

1.0 Introduction

The explosion of personal computers and major advances in the field of telecommunications were the beginning of the web as its known today. The emergence of web interconnected computers to work together and share the necessary data paved the way for Berners-Lee (1989). The growth of the World Wide Web (WWW) was impressive for the past few years. Initially, the web was for exchange of documents and data and for collaboration. It was meant to be a network of workstations where the programs and databases could share their knowledge and work together in collaboration. Information is stored in large databases kept in the servers where the programs that run the servers generate web pages dynamically. Information seekers use search engines for locating, combining, and aggregating data from the internet in their quest for information. Most of this information is encoded using the Hypertext Markup Language (HTML) and thus is difficult to manipulate on a massive scale as it is made for data representation on the web primarily for human consumption and not for machine understanding and interpretation.

This is the major disadvantage of the present Web, as finding the right information from the huge collection of web documents is becoming increasingly impossible. Current information retrieval systems are imprecise, often yielding matches to thousands of pages. The major limitations of the present web are seemingly high recall and low precision; search output and results are highly vocabulary sensitive; and, vocabulary of the query and of web resources may require multiple queries or searches if the information required is spread over several web documents. These limitations coupled with the growing volume of data and information available on the Web have initiated discussions among the members of the web community related to enhancing the present web to what Berners-Lee (2001) called a semantic web.

The primary requirement of the semantic web is that the underlying data need to be structured properly with semantic and syntactic meaning so that computers can understand the data they store. The aim of the semantic web iss to transform the current web consisting of hyperlinked pages into a web of knowledge that is machine processable. Realization of the semantic web depends on a set of web-related technologies designed to facilitate machine processing of data and interoperability. The semantic web promises to overcome the challenge of integrating and querying highly diverse and distributed resources. Systems based on the semantic web provide sophisticated frameworks to manage and retrieve knowledge.

Knowledge organization systems (KOSs) play a major role in structuring development of data for the semantic web. KOS tools such as library classification systems, thesauri, taxonomies, controlled vocabularies, terminologies, etc., can be exploited to support the development of the semantic web. Among these, library classification systems, especially faceted schemes, have been recognized as an important source of structured and formalised vocabularies that can be utilized to support the development of the semantic web. All KOSs try to bring domain knowledge formulated in a conceptual framework. Ontologies are well known knowledge organization (KO) tools, which can be utilized for capturing domain knowledge, assigning semantic meaning to information and representing data for machine consumption.

Since its inception the semantic web has gained steady acceptance in the science and technology community. Several projects have been undertaken to demonstrate the potential of the semantic web.WordNet, EuroWordNet, GUM, Mikrokosmos, and SENSUS are linguistic ontologies which use words as grammatical units. The purpose of this type of ontology is to describe semantic constructs rather than to model a specific domain. They offer quite a heterogeneous amount of resources, used mostly in natural language processing.

– WordNet (Miller et al. 1990, 1995) is a very large lexical database for English created at Princeton University and based on psycholinguistic theories. WordNet attempts to organize lexical information in terms of word meanings rather than word forms, though inflectional morphology is also considered. For example, if you search for "trees" in WordNet, you will have the same access as if you searched for "tree." WordNet 1.7 contains 121,962 words and 99,642 concepts. It is organized into 70,000 sets of synonyms ("synsets"), each representing one underlying lexical concept. Synsets are interlinked via relationships such as synonymy and antonymy, hypernymy and hyponymy (*Subclass-Of* and *Superclass-Of*), meronymy and holonymy (*Part-Of* and *Hasa*). WordNet divides the lexicon into five categories: nouns, verbs, adjectives, adverbs, and function words.
– AGROVOC (1980) is a controlled vocabulary covering all areas of interest to the Food and Agriculture Organization of the United Nations (FAO). AGROVOC is available as an RDF-SKOS linked dataset. It consists of over 32,000 concepts, available in up to 21 languages, and linked to 16 other vocabularies and resources.
– Bio2RDF (Belleau 2008) is an open-access semantic web knowledge base that provides a mashup over 19 different data sets that include the Gene Ontology, OMIM, Reactome, ChEBI, BioCyc and KEGG.
– BioGateway (Antezana 2009) is a semantic web resource that integrates the entire set of OBO Foundry ontologies (including both accepted and candidate OBO ontologies), the complete collection of annotations from the Gene Ontology Annotation (GOA) files, fragments of the NCBI taxonomy and SWISS-PROT and IntAct. This project marked the fusion of the semantic web to systems biology.
– Gene ontology (GO) (Gene Ontology 2008) is a major bioinformatics initiative to unify the representation of gene and gene product attributes across all species. GO is part of a larger classification effort, the Open Biomedical Ontologies (OBO). The ontology covers three domains: cellular component, molecular function and biological process.

Medical ontologies are developed to solve problems such as the demand for the reusing and sharing of patient data, the transmission of these data, or the need for semantic-based criteria for statistical purposes. The unambiguous communication of complex and detailed medical concepts is a crucial feature in current medical information systems. In these systems, several agents must interact in order to share results and, thus, they must use a medical terminology with a clear and non-confusing meaning. The most notable ontologies are Galen, UMLS and ON9.

- GALEN (Rector et al. 1995), developed by the non-profit organization OpenGALEN, is a clinical terminology represented in the formal and medical-oriented language GRAIL (Rector et al., 1997). This language was specially developed for specifying restrictions used in medical domains. GALEN was intended to be used with different natural languages and integrated with different coding schemata. It is based on a semantically sound model of clinical terminology known as the GALEN COding REference (CORE) model.
- UMLS (Unified Medical Language System) (2008), developed by the United States National Library of Medicine, is a large database designed to integrate a great number of biomedical terms collected from various sources such as clinical vocabularies or classifications (MeSH, SNOMED, RCD, etc.). It is structured in three parts: Metathesaurus, Semantic Network and Specialist Lexicon.

Engineering ontologies contain mathematical models that engineers use to analyze the behavior of physical systems (Gruber and Olsen 1994). These ontologies are created to enable the sharing and reuse of engineering models among engineering tools and their users. Among the various engineering ontologies, the EngMath ontologies and PhysSys are the most notable.

- EngMath (Gruber and Olsen 1994) is a set of Ontolingua ontologies developed for mathematical modeling in engineering. These ontologies include conceptual foundations for scalar, vector, and tensor quantities as well as functions of quantities, and units of measure.

Chemistry ontologies model the composition, structure, and properties of substances, processes and phenomena. Some of the chemistry ontologies developed by the Ontology Group of the Artificial Intelligence Laboratory at UPM (Universidad Politécnica de Madrid) are: Chemicals (composed of Chemical Elements and Chemical Crystals), Ions (composed of Monatomic Ions and Poliatomic Ions), and Environmental Pollutants. All of them are available in WebODE.

- Chemicals is composed of two ontologies: Chemical Elements and Chemical Crystals. These ontologies were used to elaborate METHONTOLOGY (Fernández-López et al. 1999), an ontology development methodology. The Chemical Elements ontology models knowledge of the chemical elements of the periodic table, such as what elements these are (oxygen, hydrogen, iron, gold, etc.), what properties they have (atomic number, atomic weight,electronegativity, etc.), and what combination constraints of the attribute values they have. Chemical Elements contains 16 classes, 20 instance attributes, one function, 103 instances and 27 formal axioms.
  Chemical Crystals was built to model the crystalline structure of the chemical elements. Therefore, Chemical Elements imports this ontology. The ontology contains 19 classes, eight relations, 66 instances and 26 axioms.
- Ions is built on top of Chemical Elements and is also composed of two ontologies: Monatomic Ions (which model ions composed of one atom only) and Polyatomic Ions (which model ions composed of two or more atoms). Ions contains 62 concepts, 11 class attributes, three relations and six formal axioms.
- Environmental pollutants ontology (Gómez-Pérez and Rojas, 1999) imports Monatomic Ions and Polyatomic Ions and is composed of three ontologies: Environmental Parameters, Water and Soil. The first ontology defines parameters that might cause environmental pollution or degradation in the physical environment (air, water, ground) and in humans, or more explicitly, in their health. The second and third ontologies define water and soil pollutants respectively. These ontologies define the methods for detecting pollutant components of various environments, and the maximum concentrations of these components permitted according to the legislation in force.

The objective of this paper is to provide an overview about the semantic web technologies and tools which have significant impact on knowledge integration, querying, and knowledge sharing in many domains. Insight into the currently available semantic web tools and technologies in identifying and selecting the appropriate knowledge organization systems for domains under consideration are discussed.

2.0 Semantic Web Technologies

The semantic web is a collection of technologies and standards that allow machines to understand the meaning (semantics) of information on the web. Since 2004, the field has been dominated by formal languages and technologies included under W3C recommendations. The semantic web technologies (SWT) are described below in Figure 1.

2.1 Resource Description Framework (RDF)

Resource Description Framework (RDF) is the backbone of W3C's semantic web activity. RDF is the standard for encoding metadata and other knowledge on the semantic web—providing the language for expressing the meaning of terms and concepts in a form that machines can readily process. RDF Schema is language layered on top of RDF. RDF Schema is a simple set of standard RDF resources and properties that allows creating RDF vocabularies. The data model used by RDF Schema is similar to the data model used by object-oriented programming languages like Java and allows creating classes of data (Brickley and Guha 2004). Each RDF statement is sets of triplets (Subject/resource, Predicate/Property and Object/value) (Manola and Miller 2004). The subject denotes the object the triple is describing, the predicate identifies the attribute of the subject within the statement and the object defines the value of the predicate. An example of a statement is: http://www.example.org/index.html has a creator whose value is "Swaminathan M" The subject is the URL http://www.example.org/index.html. The predicate is the word "creator." The object is the phrase "Swaminathan M." This statement can be represented graphically as shown in Figure 2.

2.2 Description Logics

Description Logics (DLs) (Baader et al. 2003) are a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formal fashion. Description Logics offer a palette of description formalisms with differing expressive power that is employed in various application domains (such as natural language processing, databases, etc.). In DL, the basic syntactic building blocks are ato-

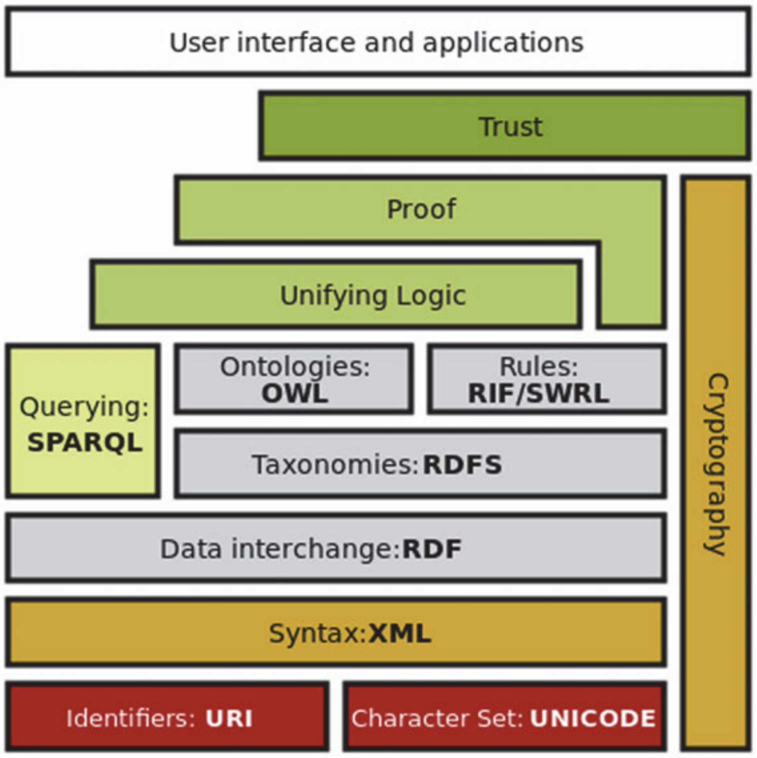


Figure 1. Semantic Web Layer Cake illustrates the architecture of the Semantic Web (Berners-Lee 2001, 20)

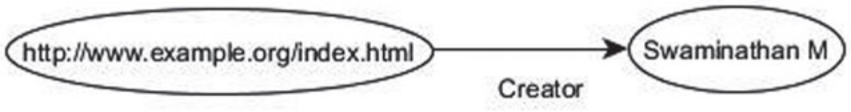


Figure 2. Graphic representation of a triple

mic concepts (unary predicates), atomic roles (binary predicates) and individuals (constants).

2.3 Web Ontology Language (OWL)

OWL is an ontology language built on top of two new semantic web standards—Resource Description Framework (RDF) and RDF Schema (RDF-S)—and designed to be compatible with the eXtensible Markup Language (XML) as well as other W3C standards. OWL has become a W3C Recommendation. OWL is primarily concerned with defining terminology which may include descriptions of classes, properties and their instances. OWL is intended to be used for greater machine interpretability of web content than that supported by XML, RDF, and RDF-S by providing additional vocabulary along with a formal semantics (Dean et al. 2004). OWL has three expressive sub-languages: OWL Full, OWL DL, and OWL Lite. All three of these languages allow describing classes, properties, and instances. OWL Lite is primarily intended for users needing a classification hierarchy and simple constraint features. OWL DL has the closest correspondence with description logics. Both OWL DL and OWL Lite require that every resource either be a class, object property, datatype property or instance. OWL Full has the same features as OWL DL, but loosens the restrictions. It is possible to treat a class as an instance, and there is no need to explicitly declare the type of each resource.

For example, an ontology that describes food science vocabulary and bibliographic datahas the basic classes and subclass relationships shown in Figure 3.

The following is a representation in OWL of Figure 3:

```
<rdf:RDF
xmlns:rdf=http://www.w3.org/1999/02/22-rdf-
   syntax-ns#””
xmlns:rdfs=http://www.w3.org/2000/01/rdf-
   schema#”
xmlns:owl=http://www.w3.org/2002/07/owl#”>
<owl:Ontology rdf:about=”xml:base”/>
<owl:Class rdf:ID=”commodity”>
food.</rdfs:comment>
<owl:intersectionOf rdf:parsetype=”Thing”>
<owl:Class rdf:about=”commodity”/>
<owl:Restriction>
<owl:onProperty rdf:resource=”#descriptors”/>
</owl:Restriction>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
<owl:Class rdf ID=”ärticle”>
<rdfs:comment>article title.</rdfs:comment>
<rdfs:subclassOf rdf:resource=”journal”/>
<rdfs:subClassof rdf:resource=”#journal name”/>
<rdfs:subClassOf>
<owl:Restriction>
<owl:OnProperty rdf:resource=””#descriptors”/>
<owl:allValuesFrom rdf:resource=”#author”/>
</owl:Restriction>
</rdfs:subClassOf>
</owl:Class>
</rdf:RDF>
```

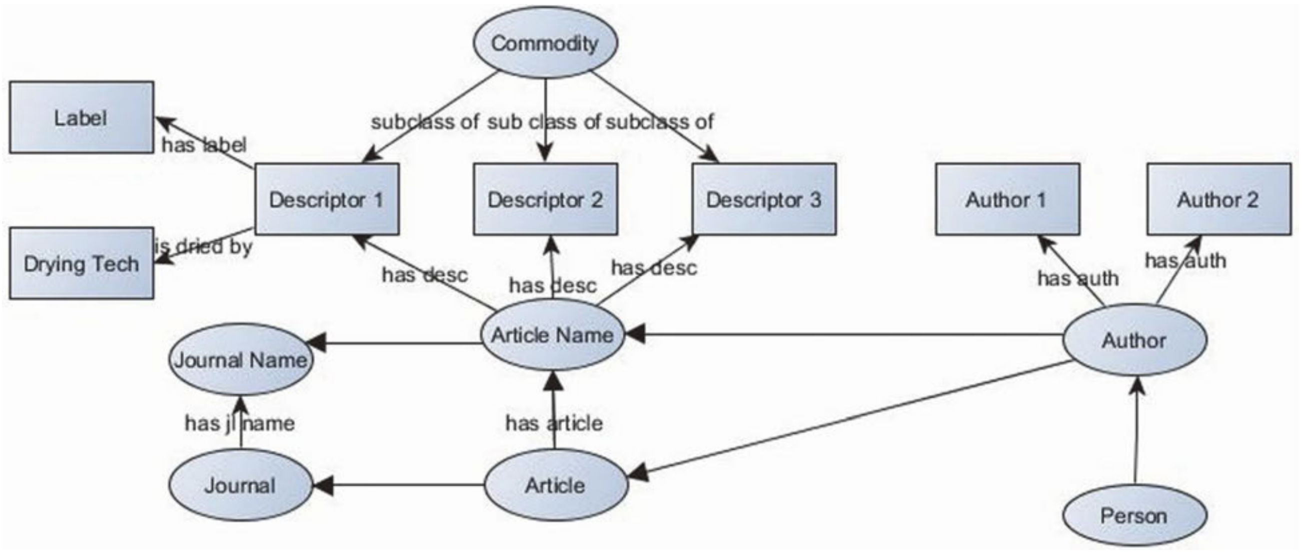


Figure 3. Basic classes and subclasses.

3.0 Ontologies

The idea of ontologies as computational artefacts has appeared in artificial intelligence (AI) and computer science. However, the term "ontology" denotes the study of existence in philosophy. In information systems, ontologies are conceptual models of what "exists" in some domains, transformed into and represented in a machine-interpretable form by using knowledge representation techniques. They are not limited to computer science and AI research, but have practical applications in a wide range of areas including medicine, geographic information systems, and biological information systems. They have been applied in diverse research areas such as knowledge engineering, knowledge representation, knowledge management, database design, natural language processing, information retrieval, etc. There are many definitions of ontologies from different researchers. One popular definition of an ontology is by Gruber (1993, 1) who defined it as an "Explicit specification of a conceptualization."

3.1 Ontology Components

Gruber (1993, 2) identified five kinds of ontology components: classes, relations, functions, formal axioms, and instances:

a. Classes represent concepts, which are considered generic entities in the broad sense;
b. Relations represent a type of association between concepts of the domain;
c. Functions are a special case of relations;
d. Formal axioms serve as model sentences that are always true. They are normally used to represent knowledge that cannot be formally defined by other ontology components; and,
e. Instances are used to represent elements or individuals in ontology.

Noy and McGuinness (2001, 3) described ontology as a formal, explicit description of concepts in a domain of discourse, including:

- Classes: the formal representation of concepts are the focus of most ontologies. Classes describe concepts in the domain, properties of each concept describing various features and attributes of the concept.

  Example:
  - A class of millet represents all kinds of millet.
  - Specific millets are instances of this class.
  - A class can have subclasses that represent concepts that are more specific than the superclass.
  - The class of all millets can be divided into foxtail millet, kodo millet, little millet, pearl millet, etc.
- Slots describe properties of classes and instances.
- Taxonomies are used to organize classes and instances in the ontology.
- Hierarchical and associative relationships are relationships between concepts. A relation represents the dependency between concepts in the domain.

An ontology can be characterized as comprising four tuples (Davies, Studer and Warren 2006, 118):

O= <C,R,I,A.>
C is a set of classes representing concepts to reason about in the given domain such as: Pests, Diseases, Prevention etc.
R is a set of relations holding between those classes, such as: relation "Harmed_By"
I is a set of instances, where each instance can be an instance of one or more classes and can be linked to other instances by relations such as: Angular, Red, Bacterial_Blight etc.
A is set of axioms such as: if plant's leaves turns yellow to brown causing defoliation, spray Carbendazim(1g).

3.2 Ontology Benefits

Noy and McGuinness (2001, 1) have proposed the following possible uses for ontology:

- To share a common understanding of the structure of information among people or software agents;
- To enable reuse of domain knowledge;
- To make domain assumptions explicit;
- To separate domain knowledge from the operational knowledge;
- To analyze domain knowledge.

The first application is one of the common goals in developing ontologies that could be targeted towards human understanding in which case conceptual graphs, topic maps or UML class diagrams could be used as representation media. The second one assumes the use of a common language for representation so others can reuse it for their domains. For example, we can reuse a general ontology, such as the UNSPSC ontology, and extend it to describe a domain of interest. The third application of ontology, to make implicit or tacit knowledge available, makes it possible to change these assumptions easily if the knowledge about the domain changes. The fourth and fifth use as proposed by Noy and McGuinness (2001) is oriented towards the architectural design and end application uses. Thus in the specific context of interoperability of information systems, the potential use of ontologies may be summarised as (Kabilan 2007, 44): "Ontology can be used as a central component of interoperability of Information Systems application to data, information, knowledge, and meta level."

3.3 Ontology Classification

There are several classifications of computer science ontologies, based on different parameters.

Uschold and Gruninger (1996, 6) have discussed in detail the principles, methods and characteristics of ontologies. They have classified ontologies into following major categories depending on formality by which a vocabulary is created.

- Highly informal: expressed loosely in natural language;
- Semi-informal: expressed in a restricted and structured form of natural language greatly increasing clarity by reducing ambiguity;
- Semi-formal expressed in an artificial formally defined language, e.g. Ontolingua version of the Enterprise Ontology;
- Rigidly formal: meticulously defined terms with formal semantics, theorems and proofs of such properties as soundness and completeness, e.g. TOVE.

Guarino (1998, 7-8) classifies them based on their level of generality in:

- Top-level ontologies describe domain-independent concepts such as space, time, object, event, etc., which are independent of specific problems;
- Domain and task ontologies describe the vocabulary related to a generic domain or a generic task or activity;
- Application ontologies describe concepts depending on a particular domain and task that are often specializations of both the related ontologies.

Gómez-Perez, Fernández-López, and Corcho (2003, 2-3) classify ontology based on the level of specification of relationships among the terms gathered on the ontology, in:

- Lightweight ontologies, which include concepts, concept taxonomies, and relationships between concepts and properties that describe concepts;
- Heavyweight ontologies, which add axioms and constraints to lightweight ontologies;
- Those axioms and constraints clarify the intended meaning of the terms involved in the ontology.

3.4 Ontology Design Principles

To guide and evaluate our designs, we need objective criteria that are founded on the purpose of the resulting artefact, rather than based on *a priori* notions of naturalness or truth. Gruber (1993) proposed a preliminary set of design criteria for ontologies whose purpose is knowledge sharing and interoperation among programs based on a shared conceptualization.

- Clarity: the concepts in an ontology should be defined in a formal way that communicates the intended meaning of defined terms. Definitions should be objective; the motivation for defining a concept might occur from social context or computational needs.
- Coherence: an important criterion for ensuring the consistence of concepts that are defined formally. It should allow inferences that are logically consistent with the defining axioms.
- Extendibility: an ontology should be designed to anticipate the uses of the shared vocabulary. It should provide a conceptual foundation for a range of certain tasks, and the representation should be crafted so that one can extend and specialise the ontology monotonically.
- Minimal encoding bias: result when a representation choice is made merely for the convenience of notation or implementation. Encoding bias should be minimized because knowledge-sharing agents may be implemented in different representation systems and styles of representation.
- Minimal ontological commitment: an ontology should require the minimal ontological commitment adequate to support the anticipated knowledge sharing activities.

These criteria have now become the basis for any ontology designer for AI and information systems. These criteria define the requirements only on the ontology artefact that is to be designed and developed. They aim only to ensure that the ontology is correct, cohesive and true. They do not reflect upon the intended purpose for the designed ontology. The domain view, that a designer adopts may be different depending on the intended use of the ontology.

3.5 Ontology Design Methodologies

The construction of ontologies may be improved if some methodology is applied. The goal of using a methodology is to obtain a good result following a set of steps which usually are based on best practices. Most of the methodologies for building ontologies are based on the experience of people involved in their construction. In several cases, methodologies are extracted from the way in which a particular ontology was built. Nowadays, methodologies are more focused on modelling knowledge than on developing applications. Thus, such methodologies are suitable alternatives for modelling knowledge instead of good alternatives for managing an information technology project centered on ontologies. Some significant ontology design and development methodologies are briefly summarized below. These specific methodologies were selected because the first two were amongst the first information-systems-oriented ontology design methodologies, and were successfully tested in developing enterprise ontologies. The third is a popular guide to developing ontologies in most widely used, open-source ontology editors. The fourth methodology is based on the Unified Software Development Process or Unified Process (UP) that gives a detailed account of the activities to be carried out similar to a software development project plan. The final methodology is based on the first two and is also based on software development principles.

3.5.1 Uschold and Gruninger's Skeletal Method

Uschold and Gruninger (1996) provide guidelines for ontology designing based on their experiences in designing the Enterprise Ontology (Uschold et al. 1995). The processes are shown in Figure 4.

– Purpose and Scope: identifying the purpose, scope and domain of an ontology to be constructed besides determining the users and developers.
– Building the Ontology: starts with three aspects capture, coding, and integration of existing ontologies.
– Capture: suggest identification of the key concepts and relationships in the domain of interest, production of precise, unambiguous text definitions for such concepts and relationships, identification of terms to refer to such concepts and relationships and finally agreeing concepts, relationships, and their names.
– Coding: includes explicit representation of the conceptualisation captured in the previous stage in some formal language. This will involve committing to the basic terms that will be used to specify the ontology (e.g. class, entity, relation), choosing a representation language and code the ontology.
– Integrating Existing Ontologies: propose the use of existing ontologies in the ontology capture or coding or both the processes.
– Evaluation: agree that evaluation of produced ontology is vital and refers to other related research done in the same domain and to adapt it for ontologies.
– Documentation: documenting the ontology process that facilitates both formal and informal documentation.

3.5.2 Gruninger and Fox Method

Gruninger and Fox (1995) proposed a more formal design approach compared to Uschold's skeletal method. This methodology is based on the experience in developing the TOVE project ontology Gruninger (1995) within the domain of business processes and activities modelling. The steps proposed are illustrated in Figure 5.

– Capture of motivating scenarios: the development of ontologies is motivated by scenarios that arise in the application. Any proposal for a new ontology or extension to ontology should describe one or more motivating scenarios and the set of intended solutions of problems presented in the scenarios.
– Formulation of informal competency questions: these are based on the scenarios obtained in the preceding

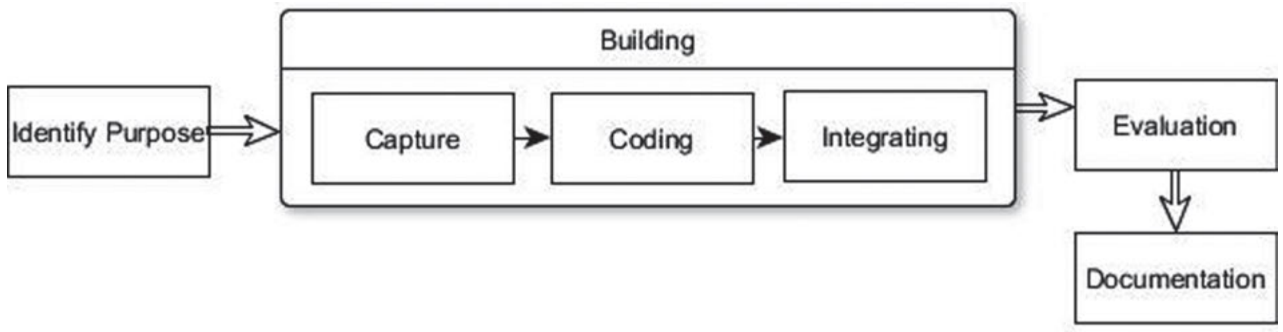


Figure 4. Main processes of the Uschold and Gruninger method.

step and can be considered as expressiveness requirements that are in the form of questions. Ontology must be able to represent these questions using its terminology, and be able to characterize the answers to these questions using the axioms and definitions.
– Specification of the terminology of the ontology within a formal language: the terminology of the ontology must be specified using first-order logic. Once the informal competency questions are available, the set of terms used can be extracted from the questions. These terms will serve as a basis for specifying the terminology in a formal language.
– Formal Competency Questions: once the competency questions have been posed informally, and the terminology of the ontology has been defined, the competency questions are defined formally.
– Specification in First-Order Logic—Axioms: axioms in the ontology specify the definitions of terms in the ontology and constraints on their interpretation; they are defined as first-order sentences using the predicates of the ontology. Axioms must be provided to define the semantics, or meaning, of these terms. This development of axioms for the ontology about the competency questions is, therefore, an iterative process.
– Completeness Theorems: once the competency questions have been formally stated, it is necessary to define the conditions under which the solutions to the questions are complete.

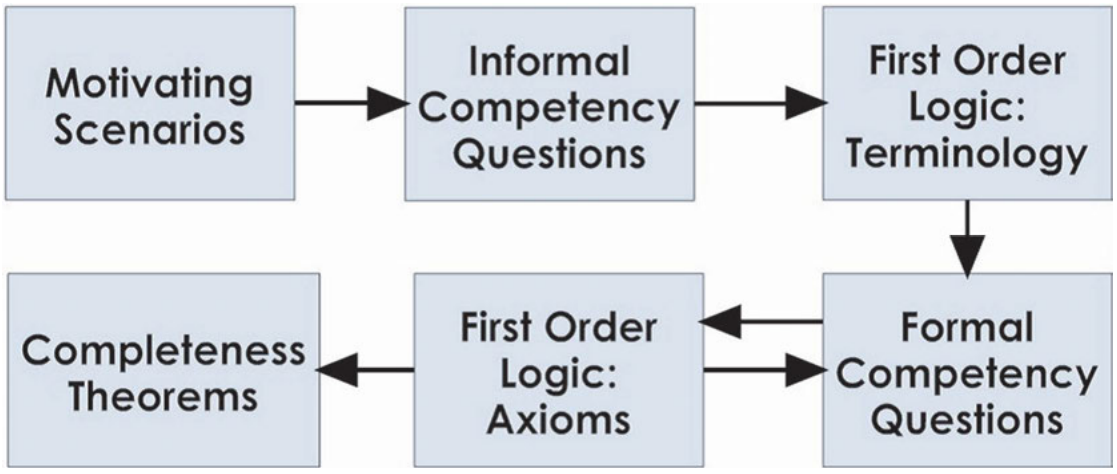
3.5.3 Noy and McGuinness Method

Noy and McGuinness' (2001) approach is more like a user manual for an ontology to be designed specifically using the Protégé ontology editor. They illustrate the process of capturing the concepts, the slots, and the role restrictions. But, on analysis, their basic design methodology is similar to that proposed by the Gruninger-Fox methodology or the Uschold-Gruninger method. Noy and McGuinness proposed a knowledge engineering method for building ontologies. They describe an iterative approach to ontology development and start with a rough first pass at the ontology. They have emphasized three fundamental rules in ontology design:

– There is no one correct way to model a domain—there are always viable alternatives;
– Ontology development is necessarily an iterative process;
– Concepts in the ontology should be close to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain.

They provide a step-by-step instruction for the user to design the ontology using the Protégé, ontology editor. The steps are described below and Figure 6 illustrates the main steps in this methodology.

– Determine the domain and scope of the ontology: suggest starting the development of ontology by defining its domain and scope. They adopt the competency questions idea as suggested by Gruninger and Fox (1995).
– Consider reusing existing ontologies: suggest considering what someone else has done and checking if we can refine and extend existing sources for a particular domain and task. Reusing existing ontologies may be a requirement if the system needs to interact with other applications that have already committed to particular ontologies or controlled vocabularies.


Motivating Scenarios
Informal Competency Questions
First Order Logic: Terminology
Formal Competency Questions
First Order Logic: Axioms
Completeness Theorems
Figure 5. Gruninger and Fox Design Methodology.

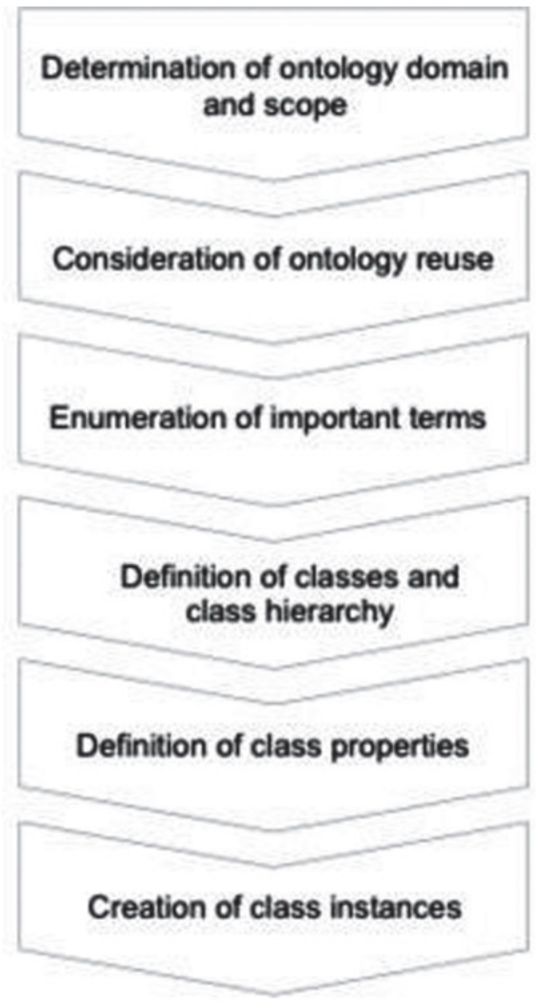

Determination of ontology domain and scope
Consideration of ontology reuse
Enumeration of important terms
Definition of classes and class hierarchy
Definition of class properties
Creation of class instances

Figure 6. Noy and McGuiness Method.

– Enumerate important terms in the ontology: they begin by identifying key concepts and terminologies relevant for the domain.
– Define the classes and the class hierarchy: they propose a combination of the top-down and bottom-up approach. The first step usually starts by defining classes whatever the approach is used. In their view, none of these three methods suggested by Uschold and Gruninger (1996) is inherently better than any of the others. The approach to take depends strongly on the personal view of the domain.
– Define the properties of classes—slots: define the classes and describe the internal structure of concepts.
– Define the facets of the slots: slots can have different facets describing the value type, allowed values, the number of the values (cardinality), and other features of the values the slot can take.
– Create instances: the last step is creating individual instances of classes in the hierarchy.

It may be noticed that the method is simple, explicit to follow for an information system designer except for the shortcoming that it is not implementation tool independent.

3.5.4 UPON (Unified Process for Ontology building)

Nicola, Missikoff and Navigli's (2009) approach is based on Unified Process for Ontology (UPON), an incremental methodology for ontology building. It stems its characteristics from the UP, one of the most widespread and accepted methods in the software engineering community, and uses the UML to support the preparation of all the blueprints of the ontology development (figure 7).

The UPON methodology has the following phases:

– Requirements workflow: capturing requirements consists in specifying the semantic needs and user view of the knowledge to be encoded in the ontology.
– The analysis workflow: the conceptual analysis concerns the refinement and structuring of the ontology requirements identified in the previous workflow.
– The design workflow: the main goal of this workflow is to give an ontological structure to the set of glossary entries gathered in the reference glossary.
– The implementation workflow: the purpose of this workflow is to encode the ontology in a rigorous, formal language.
– The test workflow: the test workflow is conceived to verify the semantic and pragmatic quality of the ontology since syntactic quality is checked in the previous workflow and social quality can be checked only after its publication.

3.5.5 METHONTOLOGY

The METHONTOLOGY framework by Fernandez, Gomez-Perez and Juristo (1997) enables the building of ontologies from scratch or from reusing other ontologies. The ontology development process is based on the IEEE standard (1996, 2006) software life cycle process for carrying out each activity. Several of the steps proposed here are similar to those of Uschold and Gruninger (1996), and Gruninger and Fox (1995). But the prominent difference is their stress on the evaluation and documentation steps. Each phase consists of activities that pass through many stages as shown in Figure 8.

– Planification: the designer should plan the entire development process like the tasks, time and resource allocation, etc.
– Specification: the goal of the specification phase is to produce either an informal, semi-formal or formal on-

tology specification document written in natural language, using a set of intermediate representations or using competency questions, respectively. This phase is similar to the competency questions phase as recommended by Gruninger and Fox (1994) and Uschold and Gruninger (1996).

– Knowledge Acquisition: independent activity in the ontology development process. They support the use of existing knowledge bases and knowledge acquisition using techniques as proposed by Uschold and Gruninger (1996). Experts, books, handbooks, figures, tables and even other ontologies are sources of knowledge from which the knowledge can he elucidated using in conjunction techniques such as brainstorming, interviews, formal and informal analysis of texts, and knowledge acquisition tools.
– Conceptualization: structures the domain knowledge in a conceptual model that describes the problem and its solution regarding the domain vocabulary identified in the ontology specification activity. The activities to be carried out are as follows:
  - Build a complete glossary of terms (GT) to identify which terms include concepts, instances, verbs and properties.
  - Group the gathered terms in GT as concepts and verbs.
  - Build concept classification tree following the guidelines as prescribed in Gomez-Perez, Fernandez, and De Vicente (1996). Verbs which represent actions in the domain are described using Vicente (1997).
  - In the data dictionary, describe and gather all the useful and potentially usable domain concepts, their meanings, attributes and instances

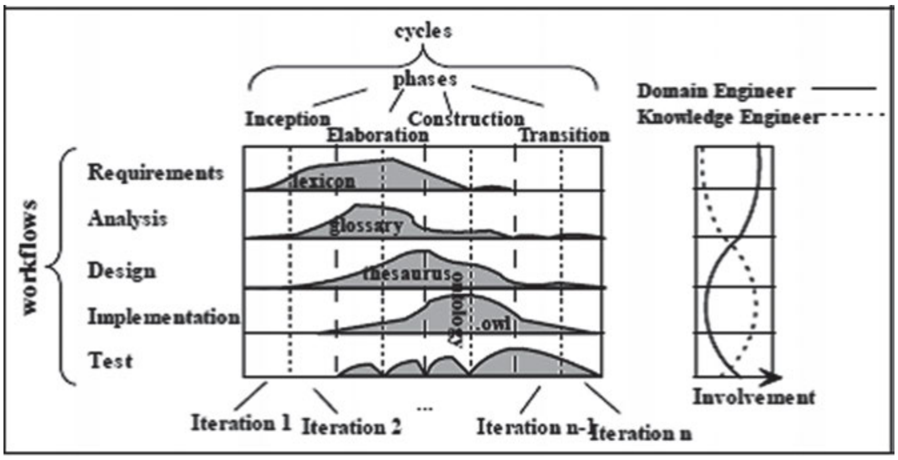


Figure 7. UPON framework (Nicola, Missikoff and Navigli 2009, 260)

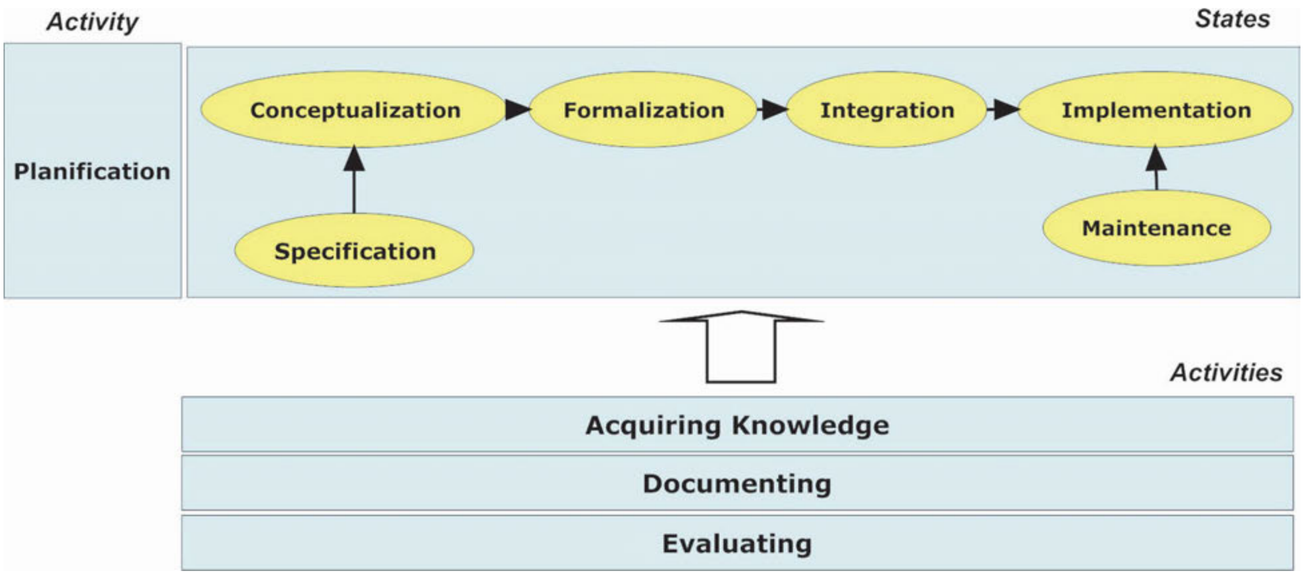


Figure 8. METHONTOLOGY: Constituent activities and their states (Uschold and Gruninger 1996, 35)

- In the verbs dictionary, express the meaning of verbs in a declarative way.

– Formalization: to transform the conceptual model into a formal or semi-compatible model. We need to formalize it using frame-oriented or description logic representation systems.
– Integration: of ontologies is required when building a new ontology. Reusing other ontologies that are already available may be considered instead of starting from scratch.
– Implementation: in this phase the ontology is codified in a formal language which is machine processable implementation language.
– Evaluation: is to carry out a technical judgment of the ontologies, their software environment, and documentation about a frame of reference during each phase and between phases of their life cycle.
– Documentation: as there are no guidelines to document ontologies because of the absence of methodologies to build ontologies, they propose documentation as an activity to be done during the whole ontology development process.
– Maintenance: any time there might be a need for including or modifying definitions in the ontology. To maintain the ontology is an important activity to be done carefully. Guidelines for maintaining ontologies are also required.

3.6 Ontology Development Tools

Most of the languages are supported by tools and six tools have been found most relevant viz., Ontolingua, WebOnto, WebODE, Protege-2000, OntoEdit and OilEd.

Ontolingua was the first ontology tool developed in the Knowledge Systems Laboratory (KSL) at Stanford University at the beginning of the 1990s. The Ontolingua system provides users with the ability to publish, browse, create, and edit ontologies stored on an ontology server. The ontology editor is the main application inside the server and works with a form- based web application. The underlying language is the Ontolingua language. It provides many of the facilities that are crucial for promoting the use of ontologies and knowledge level agent interaction such as semi-formal representation language, browsing and retrieval of ontologies from repositories (Farquhar 1997).

WebOnto is a web-based tool for visualization, browsing, and development of ontologies and knowledge models specified in OCML. It was developed by the Knowledge Media Institute at the Open University as part of several European research projects in the late 90s. WebOnto is a Java client connected to a customized web server. It adopts the knowledge model of OCML and distinguishes between domain, tasks, problem-solving methods and applications. Its main advantage over other available tools is that it supports editing ontologies collaboratively, allowing synchronous and asynchronous discussions about the ontologies being developed (Domingue and Motta 1999).

WebODE is scalable ontological engineering on the web. It has been developed by the Ontology and Knowledge Reuse Group, at the Technical University of Madrid. It has three-tier architecture: the user interface, the application server, and the database management system. WebODE's ontology editor allows the collaborative edition of ontologies at the knowledge level, supporting the conceptualization phase of METHONTOLOGY and most of the activities of the ontology's life cycle (reengineering, conceptualization, implementation, etc.). The workbench is built on an application server basis, which provides high extensibility and usability by allowing the addition of new services and the use of existing services. WebODE's knowledge model is extracted from the set of intermediate representations of METHONTOLOGY such as concepts, groups of concepts, relations, constants and instances. (Arpírez 2003).

Protegé is a free, open-source ontology editor that assists users in the construction of large electronic knowledge bases. It was developed by Stanford Medical Informatics (SMI) at Stanford University. It was written in Java and runs on a wide variety of operating systems. The Protégé system has dozens of plugins. These plugins provide alternative visualization mechanisms, enable management of multiple ontologies, including merging and version management, allow the use of various inference engines and problem solvers with Protégé ontologies, and provide other functionalities. It also provides a Java API for application developers to access and modify all aspects of Protégé knowledge bases and its user interface. Protégé stores ontologies in many different formats including relational databases, OWL, XML, RDF, and HTML. It supports the latest OWL 2 Web Ontology Language and RDF specifications from the World Wide Web Consortium.

OntoEdit has been developed by AIFB in Karlsruhe University (Sure et.al. 2002). It is similar to the previous tools. It is an extensible and flexible environment, based on a plugin architecture, which provides functionality to browse and edit ontologies. It supports multilingual development, and the knowledge model is related to frame-based languages.

OilED supports the construction of OIL-based ontologies. The basic design has been influenced by similar tools such as Protégé, OntoEdit. It integrates a reasoner (FaCT) and extends the expressive power of other frame-based tools, but ignores services and flexibility completely. OilEd is a demonstration tool (Bechhofer 2001).

After a detailed study on tools, a comparison is presented with respect to their features. Table 1 shows comparison of the ontology tools.

3.7 Reasoning Tools

A semantic reasoner, reasoning engine, rules engine, or simply a reasoner, is a piece of software able to infer logical consequences from a set of asserted facts or axioms. The notion of a semantic reasoner generalizes that of an inference engine. The inference rules are commonly specified using an ontology language, and often a description language. There are many reasoners which use first-order predicate logic to perform reasoning. Forward chaining and backward chaining are the strategies of ontology reasoners (Wikipedia 2015). In forward chaining an inference engine searches the inference rules until it finds one where the IF clause is known to be true. When found it can conclude, or infer, the THEN clause, resulting in the addition of new information to its dataset. It starts with some facts and applies rules to find all possible conclusions. The forward-chaining method of inference increases storage size and overhead associated with insertion and removal operations in an attempt to improve retrieval performance in a knowledge base. In backward chaining an inference engine would search the inference rules until it finds one which has a THEN clause that matches the desired goal. If the IF clause of that inference rule is not known to be true, then it is added to the list of goals (in order for goal to be confirmed it must also provide data that confirms this new rule). This approach starts with the desired conclusion and works backward to find supporting facts. (Hebeler et al. 2009).

Among the large number of reasoners available, the popular reasoners suited for protégé are Pellet, RACER, FACT++, Snorocket, HermiT, CEL, ELK, SWRL-IQ and TrOWL.

Pellet is an open source Java-based OWL-DL reasoner developed by The Mind Swap group. It is the first reasoner to support all of OWL-DL, i.e. the Description Logic (DL) SHOIN (D), and has been extended to OWL 2 DL SROIQ(D). It is implemented in Java and is open sourced under a liberal license. Pellet is a Description Logic reasoner based on tableaux algorithms and incorporates novel optimization technique for incremental reasoning against dynamic knowledge bases. It uses the type system approach to support reasoning with datatypes. It reasons ontologies through Jena as well as OWL-API interfaces and also supports the explanation of bugs. (Sirin et al. 2007).

Racer also known as RACERPro is a reasoner for OWL DL and it was the first OWL reasoner. The RacerPro system is tailored for supporting ontology-based application which mainly build on the exploitation of assertional rea-

Feature	**Ontolingua**	**Webonto**	**Webode**	**Protégé**	**Ontoedit**	**Oiled**
Developers	Stanford Univ.	Knowledge Media Inst	Univ. of Madrid	Stanford Univ.	Ontoprise	University of Manchester
Availability	Open Source	Open Source	Open Source	Open Source	Software License	Freeware
Semantic Web Architecture	Standalone & Client Server	Client Server	Application Server	Standalone & Client Server	Eclipse client/server	Standalone
Extensibility	-	-	Plug-ins	Plug-ins	Plug-ins	Plug-ins
Backup Management	No	No	No	No	No	No
Ontology Storage	File & DBMS	File	File	File & DBMS	DBMS	File
Import from languages	RDF(S), OWL, HTML, XML, RDF	RDF(S), OWL, HTML, XML, RDF	XML, RDF(S), OIL, DAML+OIL, OWL, CARIN, FLogic, Jess, Prolog	RDF(S), OWL, HTML, XML, RDF	XML(S), OWL, RDF(S),	RDF(S), OIL, DAML+OIL, and the SHIQ XML format.
Export to languages	RDF(S), OWL, HTML, XML, RDF, F-Logic	RDF(S), OWL, HTML, XML, RDF	XML, RDF(S), OIL, DAML+OIL, OWL, CARIN, FLogic, Jess, Prolog	RDF(S), OWL, HTML, XML, RDF, F-Logic	OWL, RDF(S), F-Logic, Excel	DAML+OIL, RDF(S),OWL
Axiom language	Yes(PAL)	-	Yes(WAB)	Yes(PAL)	F-Logic	Yes
Ontology libraries	Yes	-	No	Yes	Yes	-

Table 1. Comparison of the ontology tools

soning (Abox reasoning). It was the first system which efficiently supported concrete domains for Tbox and Abox reasoning and later extended to also support inverse roles and qualitative number restrictions as part of the description logic (DL) SHIQ, a practically relevant subset of OWL. (Haarslev et al. 2011).

Fact++. A description logic reasoner implements a tableaux decision procedure for the well known SHOIQ description logic, with additional support for datatypes, including strings and integers. FaCT++ is implemented using C++ to create a more efficient software tool, and to maximise portability. The latest version of FaCT++ supports OWL and is based on the description logic SROIQ. A tableau-based decision procedure is implemented for general TBoxes and incomplete support for ABoxes. (Tsarkov and Horrocks 2006).

Snorocket for Protegé is a Java implementation of the polynomial classification algorithm described by Baader, Lutz and Suntisrivaraporn (2006) for the lightweight description logic EL**++** and packaged for use as a reasoner in Protegé.

Hermit is a new reasoner for SHOIQ+ (and OWL) based on novel algorithms and optimizations (Shearer et al. 2008). It is available as an open-source Java library and includes both a Java API and a simple command-line interface. It can process ontologies in any format handled by the OWL API, including RDF/XML, OWL Functional Syntax, KRSS, and OBO. HermiT shows significant performance advantages over other reasoners across a wide range of real-world ontologies. In several cases, HermiT can classify ontologies that no other reasoner can process. It also includes support for some non-standard ontology features, such as description graphs.

CEL is known for its scalability of reasoning in the lightweight DL EL++, which has been proved suitable for several ontology applications (Baader, Lutz and Suntisrivaraporn 2006). Recently, the DL EL++ has been adopted as the logical underpinning of the OWL 2 EL profile of the new Web Ontology Language. To integrate the reasoner to the OWL user community, they have implemented the OWL API for CEL. This shows CEL's reasoning capabilities to Protegé users.

ELK is a specialized reasoner for the lightweight ontology language OWL EL. The practical utility of ELK is in its combination of high performance and comprehensive support for language features. It employs a consequence-based reasoning engine that can take advantage of multi-core and multi-processor systems. A modular architecture allows ELK to be used as a stand-alone application, Protégé plug-in or programming library (either with or without the OWL API) (Kazakov, Krötzsch and Simančík 2012).

SWRL-IQ (SWRL Inference and Query Tool allows users to create, edit, save, and submit queries to an underlying inference engine based on XSB Prolog. Some distinct features from other reasoning tools are goal-oriented backward-chaining reasoning, flexible constraint handling that allows for very declarative rules and queries, powerful SWRL extensions, and tracing and debugging features for the explanation of reasoning results. It is implemented in a flexible way to allow for different syntax front ends and reasoning back ends (Daniel and Riehemann 2012).

TrOWL is a tractable reasoning infrastructure for OWL 2, which comes with a family of ontology languages. It contains a profile checker to detect which profile an ontology may already fit into, and has support for heavyweight reasoning using a plug-in reasoner such as Fact++, Pellet, Hermit, or Racer. TrOWL is based around two primary technologies: Language transformations, and lightweight reasoners (Thomas, Pan and Ren 2010).

A comparison of ontology reasoners is presented with respect to their attributes in Table 2.

4. RDF Storage and Retrieval Systems

Ontologies are often used to improve data access. For this purpose, existing data must be linked to an ontology and appropriate access mechanisms have to be provided. Ontologies are expressed in different query languages (RDF, OWL, DAML, etc.) and stored in different types of repositories (databases, text files, URLs, etc.), there is a need to access the semantic content in a common way for all the applications. Different ontology frameworks implement different APIs to access ontologies (Sesame, Jena, etc.). The most common approach for accessing ontology-based data is via an RDF storage and retrieval technologies.

4.1 Jena Semantic Web Framework

Jena is a Java framework for building semantic web applications from Apache Software (2011). It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. It is an open source project and its development started with HP Labs Semantic Web Program. The Jena Framework includes RDF API, Reading and writing RDF in RDF/XML, N3 and N-Triples, an OWL API, In-memory and persistent storage, RDQL- a query language for RDF and SPARQL query engine, SPARQL server which can present RDF data and answer SPARQL queries over HTTP.

Jena has the graph as its core interface around which the other components are built. The main feature is its rich Model API for manipulating RDF graphs. Using the API one can choose to store RDF graphs in memory or

	Pellet	RACER	FACT++	Snorocket	HermiT	CEL	ELK	SWRL-IQ	TrOWL
License	DULI:AGPL	Own	GLGPL	Own	GLGPL	Apache	Apache	-	DULI:AGPL
Availability	Open source	Commercial	Open source	Commercial	Open source	Open source	Open source	-	Commercial
Methodology	Tableau based	Tableaux based	Tableau based	Completion rules	Hypertableau based	Completion rules	Consequence based	SWRL rules	Completion rules
Soundness	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Completeness	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Expressivity	SROIQ(D)	SHIQ	SROIQ(D)	EL+	SROIQ(D)	EL+	EL	-	SROIQ
Native Profile	DL,EL	DL	DL	EL	DL	EL	EL	-	DL,EL
Rule Support	Yes(SWRL)	Yes(SWRL)	No	No	Yes(SWRL)	No	Yes(Own rule)	Yes(SWRL)	No
Platforms	all	all	all	all	all	Linux	all	all	all
ABOX Reasoning	Yes	Yes	Yes	No	Yes	Yes	No	Yes	Yes
OWL API	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
OWL Link API	Yes	Yes	Yes	No	Yes	Yes	-	No	No
Protégé Support	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Jena Support	Yes	No	No	No	No	No	-	No	Yes
Implementation Language	Java	LISP	C++	Java	Java	LISP	Java	Prolog	Java

Table 2. Comparison of the reasoners.

in persistent stores. Jena uses ARQ query engine that supports the SPARQL (pronounced "Sparkle") RDF Query Language. It also offers an RDF Triple Store facility with SPARQL interface to be used on top of other database systems.

The Jena package contains also other systems; among those, there is Fuseki, an SPARQL Server offering an SPARQL endpoint on the top of any of the mentioned systems. An older equivalent of Fuseki is the Joseki SPARQL Server. It can run as an operating system service, as a Java web application (WAR file), and as a standalone server. It provides security (using Apache Shiro) and has a user interface for server monitoring and administration. It provides the SPARQL 1.1 protocols for query and update as well as the SPARQL Graph Store protocol.

4.2 Ontology Query Languages

RDF query language is used to get information out of a knowledge base and manipulate stored data in RDF format (Bailey, Bry, Furche and Schaffert 2005). The end users and developers can write desired queries and use the query results across a broad range of information on the Web. Several languages have been proposed for querying RDF documents, and SPARQL is introduced as a standard query language for RDF documents by W3C.

Several query languages such as RQL (RDF Query Language), SeRQL (Sesame RDF Query Language), SquishQL, RDFPath, Versa, TRIPLE, DAML+OIL Query Language, RDQL, RDFQL, N3, iTQL, RStar, SPARQL, etc., have been introduced for RDF documents. All of these query languages were intended to provide a proper query language for RDF documents. Some of the major ones are described below.

OWL Query Language (OWL-QL) is a formal language and protocol for a querying agent and an answering agent to use in conducting a query-answering dialogue using knowledge represented in the Ontology Web Language (OWL) (Fikes et al. 2003). It is an updated version of the DAML Query Language (DQL) developed by the Joint United States/European Union ad hoc Agent Markup Language. OWL-QL is intended to be a candidate standard language and protocol for query-answering dialogues among Semantic Web computational agents during which answering agents i.e. servers may derive answers to questions posed by querying agents i.e. clients. It is designed to be ideal and easily adaptable to other declarative formal logic representation languages, including, in particular, first-order logic languages such as KIF and the earlier W3C languages, RDF, RDF-S, and DAML+OIL.

RDQL (RDF Data Query Language) is a query language for RDF in Jena models. It provides a data-oriented query model so that there is a more declarative approach to complement the fine-grained, procedural Jena API. As it is "data-oriented" it only queries the information held in the models; there is no inference being done. The RDQL system only takes the description of what the application wants, in the form of a query, and returns that information, in the form of a set of bindings (W3C, 2004). RDQL is an implementation of the SquishQL RDF query language, which itself is derived from rdfDB. This class of query languages regards RDF as triple data, without schema or ontology information unless explicitly included in the RDF source.

RQL (RDF QUERY LANGUAGE) is still the only declarative language for querying both explicitly stated triples of RDF/S graphs and inferred ones by transitivity of subsumption and type relationships. It is a typed, functional language with limited recursion which relies on a formal model for RDF/S graphs permitting the interpretation of instances using one or more schema vocabularies (Karvounarakis et al. 2002). RQL adapts the functionality of semistructured/XML query languages to the peculiarities of the RDF/S data model but, it integrates smoothly RDF/S reasoning with querying (called /semantics-aware querying/). It provides sophisticated /pattern matching facilities/ under the form of generalized path expressions (GPEs) featuring variables on both labels for nodes (i.e., classes) and edges (i.e., properties).

SPARQL is an RDF query language and data access protocol for the semantic web. Its name is a recursive acronym that stands for SPARQL Protocol and RDF Query Language. It was standardized by W3C's SPARQL Working Group in 2008. The W3C Recommendation of SPARQL consists of a query language, an XML format in which query results will be returned, and a protocol of submitting a query to a query processor service remotely. The advantage of having a query language such as SPARQL are: to query RDF graphs to get specific information; to query a remote RDF server and to get streaming results back; to run automated regular queries again RDF dataset to generate reports; to enable application development at a higher level, i.e., application can work with SPARQL query results, not directly with RDF statements. (Eric et al. 2006).

For example, a basic SPARQL query can be written as follows:

```
SELECT ?title
FROM <http://example.org/book/book1>
WHERE
{
<http://example.org/book/book1>
   <http://purl.org/dc/elements/1.1/title> ?title
}
```

This query is composed of a SELECT clause identifying the variables to appear in the query results, a FROM clause indicating the dataset to be queried, and the WHERE clause providing the basic graph pattern matches the data graph. Variables in SPARQL start with a "?" or a "$". The graph pattern of this example above is simple and consists of a single triple pattern with a single variable "?title" in the object position. Only the bindings for this variable will be returned.

5.0 Conclusion

The paper provides a clear insight about the semantic web technologies, tools and languages. RDF, SPARQL, triple store and ontology facilitate the integration and analysis of heterogeneous multi-disciplinary data. Knowledge representation (KR) paradigms underlying all these technologies and languages are diverse and are based on combinations of several formalisms. We have tried to show the most important features of each of these technologies.

References

AGROVOC Multilingual Agricultural Thesaurus. 1980. AIMS. http://aims.fao.org/vest-registry/vocabularies/agrovoc-multilingual-agricultural-thesaurus

Antezana, Erick, Ward Blonde, Michel Egana, Alistair Rutherford, Alistair, Robert Stevens, Bernard De Baets, Vladimir Mironov and Martin Kuiper. 2009. "BioGateway: A Semantic Systems Biology Tool for the Life Sciences." *BMC Bioinformatics* 10 no. 10: S11.

Arpírez, Julio C, Oscar Corcho, Mariano Fernández-López, and Asuncion Gómez-Pérez. 2003. "WebODE in a Nutshell." *AI Magazine* 34 no. 3: 37-48.

Baader, Franz, Carsten Lutz and Boontawee Suntisrivaraporn. 2006. "CEL-A Polynomial-time Reasoner for Life Science Ontologies." In *Automated Reasoning*, IJCAR 2006, ed. U. Furbach and N. Shankar. Lecture notes in computer science 4130. Berlin: Springer, 287-91.

Bailey, James, François Bry, Tim Furche and Sebastian Schaffert. 2005. "Web and Semantic Web Query Languages: A Survey." *http://www.en.pms.ifi.lmu.de/publications/PMS-FB/PMS-FB-2005-14.pdf*

Bechhofer, Sean, Ian Horrocks, Carole A. Goble, and Robert Stevens. 2001. "OilEd: A Reasonable Ontology Editor for the Semantic Web." In *Description Logics,* ed. Carole A. Goble, Deborah L. McGuinness, Ralf Möller and Peter F. Patel-Schneider. CEUR Workshop Proceedings 49. Berlin: Springer, 396-408.

Belleau, Francois, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault and Jean Morissette. 2008. "Bio2RDF: Towards a Mashup to Build Bioinformatics

Knowledge Systems." *Journal of Biomedical Informatics* 41: 706-16.

Berners-Lee, Tim, Ora Lassila and James Hendler. 2001. "The Semantic Web." *Scientific American* 284 no. 5: 34-43.

Brickley, Dan and Ramanathan V. Guha. 2004. "RDF Vocabulary Description Language 1.0: RDF Schema." W3C Recommendation. http://www.w3.org/TR/2014/REC-rdf-schema-20140225/

Baader, Franz, Ian Horrocks and Ulrike Sattler. 2005. "Description Logics as Ontology Languages for the Semantic Web." In *Mechanizing Mechanical Reasoning: Essays in honor of Jörg H. Siekmann on the Occasion of his 60th Birthday,* ed. Dieter Hutter and Werner Stephan. Lecture notes in computer science 2605. Berlin: Springer, 228-248.

Dagobert, Soergel. 1999. "The Rise of Ontologies or the Reinvention of Classification." *Journal of the American Society for Information Science* 50: 1119-20.

Daniel, Elenius and Susanne Riehemann. 2012. "SWRL-IQ." http://protegewiki.stanford.edu/wiki/SWRL-IQ

Davies, John, Rudi Studer and Paul Warren. 2006. *Semantic Web Technologies: Trends and Research in Ontology-Based Systems*. London: John Wiley & Sons.

Dean, Mike, Guus Schreiber, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider and Andrea L. Stein. 2004. "OWL Web Ontology Language Reference." W3C Recommendation. http://www.w3.org/TR/2004/REC-owl-ref-20040210/

Domingue, John and Enrico Motta. 1999. "Knowledge Modeling in Webonto and OCML." http://kmi.open.ac.uk/projects/ocml/

Eric, Prud'hommeaux and Andy Seaborne. 2006. "SPARQL Query Language for RDF." https://www.w3.org/TR/2006/CR-rdf-sparql-query-20060406/

Farquhar, Adam, Richard Fikes and James Rice. 1997. "The Ontolingua Server: A Tool for Collaborative Ontology Construction." *International Journal of Human Computer Studies* 46: 707-27.

Fernandez, Mariano, Asuncion Gomez-Perez and Natalia Juristo. 1997. "METHONTOLOGY: From Ontological Art Towards Ontological Engineering." *AAAI Technical Report* SS-97-06, pp. 33-40.

Fikes, Richard, Patrick Hayes and Ian Horrocks. 2004. "OWL-QL—A Language for Deductive Query Answering on the Semantic Web." *Web Semantics: Science, Services and Agents on the World Wide Web* 2 no. 1: 19-29.

The Gene Ontology Consortium. 2008. "The Gene Ontology Project." Nucleic Acids Research *36 suppl. 1: D440-4.* doi*:10.1093/nar/gkm883*

Gómez-Pérez, Asuncion and Rojas-Amaya, Ma Dolores. 1999. "Ontological Reengineering and Reuse." In *11th European Workshop on Knowledge Acquisition, Modeling and Management (EKAW'99), Dagstuhl Castle, Germany,* ed. Dieter Fensel and Rudi Studer. LectureNotes in Artificial Intelligence 1621. Berlin: Springer, 139-56.

Gómez-Pérez, Asuncion, Mariano Fernández-López and Oscar Corcho. 2003. *Ontological Engineering*. London, Springer-Verlag.

Gruber, Thomas R. 1993. "A Translation Approach to Portable Ontology Specifications." *Knowledge Acquisition* 5: 199-220.

Gruber, Thomas R and Gregory Olsen. 1994. "An Ontology for Engineering Mathematics." In*Fourth International Conference on Principles of Knowledge Representation and Reasoning, Bonn, Germany,* ed. Jon Doyle, Piero Torasso and Erik Sandewall. San Francisco: Morgan Kaufmann, pp. 258–269.

Grüninger, Michael and Mark Fox. 1995. "Methodology for the Design and Evaluation of Ontologies." *IJCAI'95, Workshop on Basic Ontological Issues in Knowledge Sharing, April 13, 1995, Montreal.* Berlin: Springer, 1-10.

Haarslev, Volker, Kay Hidde, Ralf Moller and Michael Wessel. 2011. "The RacerPro Knowledge Representation and Reasoning Systems." *Semantic Web Journal* https://www.franz.com/agraph/cresources/white_papers

Hebeler, John, Mathew Fisher, Ryan Blace and Andrew Perez-Lopez. 2009. *Semantic Web Programming*. New Jersey: Wiley Publishing.

Hjørland, Birger. 2007. "Semantics and Knowledge Organization." *Annual Review of Information Science and Technology* 41: 367-405.

Kabilan, Vandana. 2007. Ontology for Information Systems (O4IS) Design Methodology: Conceptualizing, Designing and Representing Domain Ontologies. PhD dissertation KTH Royal Institute of Technology, Stockholm.

Karvounarakis, Gregory, Alexaki Sofia, Christophides Vassilis, Plexousakis Dimitris and Michel Scholl. 2002. "RQL: A Declarative Query Language for RDF." *European projects C-Web (IST-1999-13479) and Mesmuses (IST-2000-26074), WWW2002, May 7–11, Honolulu, Hawaii, USA*. New York: ACM, 592-603.

Kazakov, Yevgeny, Markus Krötzsch and František Simančík. 2012. "ELK Reasoner: Architecture and Evaluation." *Proceedings of the 1st International Workshop on OWL Reasoner Evaluation (ORE-2012),* ed. Ian Horrocks, Mikalai Yatskevich and Ernesto Jiménez-Ruiz. CEUR Workshop 858.

Manola, Frank and Eric Miller. 2004. "RDF Primer." W3C Recommendation. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

Miller, George A., Richard Beckwith, Christiane Fellbaum, Derek Gross and Katherine Miller. 1990. "Introduction to WordNet: An On-line Lexical Database." *International Journal of Lexicography* 3: 235-44.

Miller, George A. 1995. "WordNet: A Lexical Database for English." *Communications of the ACM* 38 no. 11: 39-41.

Nicola, Antonio De., Michele Missikoff and Roberto Navigli. 2009. "A Software Engineering Approach to Ontology Building." *Information Systems* 34: 258-75.

Noy, Natalya F and Deborah L. McGuinness. 2001. "Ontology development 101: A Guide to Creating your First Ontology." http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf

Rector, Alan L., William D. Solomon, William A. Nowlan and T. William Rush. 1995. "A Terminology Server for Medical Language and Medical Information Systems." *Methods of Information in Medicine* 34: 147-57.

Rector, Alan L, Sean Bechhofer, Carole A. Goble, Ian Horrocks, William A. Nowlan and William D. Solomon. 1997. "The GRAIL Concept Modelling Language for Medical Terminology." *Artificial Intelligence in Medicine* 9: 139-71.

Sirin, Evren, Bijan Parsia, Bernardo C. Grau, Aditya Kalyanpur and Yarden Katz. 2007. "Pellet: A Practical OWL-DL Reasoner." *Web Semantics: Science, Services and Agents on the World Wide Web* 5 no. 2: 51-53.

Smith, Michael. K, Deborah L. McGuinness and Chris Welty. 2004. "OWL Web Ontology Language Guide W3C Recommendation." http://www.w3.org/TR/2004/REC-owl-guide-20040210/

Sure, York and Victor Losif. 2002. "First Results of a Semantic Web Technologies Evaluation." *Proceedings of the Common Industry Program at the federated event co-locating the three international conferences: DOA'02: Distributed Objects and Applications; ODBASE'02: Ontologies, Databases and Applied Semantics; CoopIS'02: Cooperative Information Systems, Irvine*, California, 1-10.

Thomas, Edward, Jeff Z. Pan and Yuan Ren. 2010. "TrOWL: Tractable OWL 2 Reasoning Infrastructure." In *The Semantic Web: Research and Applications. Extended Semantic Web Conference 2010,* ed. L. Aroyo. Lecture Notes in Computer Science 6089. Berlin: Springer.

Tsarkov, Dmitry and Ian Horrocks. 2006. "FaCT++ Description Logic Reasoner: System Description." staff.cs.manchester.ac.uk/~tsarkov/papers/TsHo06a.pdf

National Library of Medicine. 2008. "Unified Medical Language System (UMLS)." https://www.nlm.nih.gov/research/umls/new_users/online_learning/OVR_001.html

Uschold, Mike and Michael Gruninger. 1996. "Ontologies: Principles Methods and Applications." *The Knowledge Engineering Review* 11: 93-155.

Veltman, Kim H. 2001. "Syntactic and Semantic Interoperability: New Approaches to Knowledge and the Semantic Web." *The New Review of Information Networking* 7: 159-83.

Veltman, Kim H. 2002. "Challenges for a Semantic Web." *Proceedings of the International Workshop on the Semantic Web 2002 (at the Eleventh International World Wide Web Conference), Honolulu, Hawaii, May 7, 2002.* http://semanticweb2002.aifb.uni-karlsruhe.de/proceedings/Position/veltmann.pdf

Veltman, Kim H. 2004. "Towards a Semantic Web for Culture." *Journal of Digital Information* 4, no. 4. https://journals.tdl.org/jodi/index.php/jodi/article/view/113