A·L·I·C·E
Adaptive Learning via Intuitive/Interactive Collaborative and Emotional systems

Project Number:	**257639**
Project Title:	ALICE: ADAPTIVE LEARNING VIA INTUITIVE/INTERACTIVE, COLLABORATIVE AND EMOTIONAL SYSTEMS
Instrument:	Specific Targeted Research Projects
Thematic Priority:	ICT-2009.4.2:Technology-Enhanced Learning
Project Start Date:	June 1st, 2010
Duration of Project:	24 Months
Deliverable:	**D3.1.2: Methodologies and Techniques for Knowledge Modeling and Representation v2**
Revision:	2.0
Workpackage:	WP3: Live and Virtualized Collaboration
Dissemination Level:	Public
Due date:	November 30, 2011
Submission Date:	November 30, 2011
Responsible	UOC
Contributors:	UOC

PROJECT CO-FUNDED BY THE EUROPEAN COMMISSION WITHIN THE SEVENTH FRAMEWORK PROGRAMME (2007-2013)

SEVENTH FRAMEWORK PROGRAMME

Version History			
Version	**Date**	**Changes**	**Contributors**
0.1	30/09/2010	Initial version	Jordi Conesa (UOC)
0.2	31/10/2010	Changes in structure. Add Section 4	Jordi Conesa (UOC)
0.3	15/11/2010	Changes in structure. Enrich Section 4 and add Section 5	Jordi Conesa (UOC)
0.4	30/11/2010	Complete Section 2 - 5	Jordi Conesa (UOC)UOC
0.5	15/12/2010	Feedback from internal review	Santi Caballé (UOC)
0.6	31/12/2010	Complete all sections.	Jordi Conesa (UOC)
0.7	24/01/2011	Feedback from external review	TUG, CRMPA
0.8	30/01/2011	Add proposed changes 1	Jordi Conesa, David Gañan (UOC)
0.9	18/02/2011	Add proposed changes 2	Santi Caballé (UOC)
1.0	28/02/2011	Final document	Santi Caballé (UOC)
1.1	01/10/2011	Rewrite the CS2 ontology (Section 5.1.1)	Jordi Conesa (UOC)
1.2	10/10/2011	Add information on CSML conversion and alignment CS2 and SIOC (Section 5.1.2)	Jordi Conesa (UOC)
1.3	10/11/2011	Add information on populating the CS2 ontology from Forum (Section 5.1.3)	Jordi Conesa (UOC)
1.4	15/11/2011	Add new section on extending CS2 with sentiment and opinion information (Section 5.1.4)	Jordi Conesa (UOC)
1.5	17/11/2011	Add new section on validation of the CS2 ontology (Section 5.3)	Jordi Conesa (UOC)
1.6	18/11/2011	Completed document Version 2	Santi Caballé

			(UOC)
1.7	28/11/2011	Feedback from external reviewers	TUG, CRMPA
1.8	29/11/2011	Development of proposed changes 1	Jordi Conesa, (UOC)
1.9	29/11/2011	Development of proposed changes 2	Jordi Conesa (UOC)
2.0	30/11/2011	Final document	Santi Caballe (UOC)

Table of Contents

1 Introduction

This report describes activities of Work package 3, Task 3.2 of the ALICE project. The aim of ALICE is to build an adaptive and innovative environment for e-learning. To this end, personalization, collaboration, and simulation aspects are combined and also affective and emotional aspects are considered. In particular, two specific contexts will be considered in ALICE: university instruction and training about emergency and civil defense.

1.1 Purpose

As the task of WP 3 is to support live and virtualized collaboration sessions, its main objective is to define methodologies and techniques to use the Semantic Web models, languages and tools for knowledge modeling and representation in order to facilitate the virtualization process of collaborative sessions. The objective of WP 3 also includes the validation of the methodologies and techniques used, both theoretically (verify that the created ontologies are consistent and complete) and practically (verify that the presented framework really supports the virtualization process of collaborative sessions). Therefore, the aim of this document is to outline and summarize findings of research with respect to representing and supporting collaboration sessions on e-learning and to present a ontological framework to do so. In particular, we will present the state-of-the-art research based on a broad literature review of recent research in Computer Science, the framework defined to support virtual collaborative sessions and the validation of the presented framework by experimentation of the framework in real environments.

The created framework contains an ontology that defines the basic information about collaboration activities. This ontology aligns with SIOC, FOAF and SKOS in order to facilitate the importation of collaboration data specified using SIOC format and vice versa. The framework provides a converter that facilitates importing data from any kind of web forum to the ontology. Apart from collaboration information, a possible extension of the framework is presented in order to represent also the opinion and the sentiments of students during collaboration. As the experimentation has proven, the framework fully covers the requirements stated in WP3. The experiment shown that the framework allows representing information about collaboration that occurs during learning experiences, and allows taking profit of these information in order to create advanced services, such as the creation of videos that reproduce the collaboration taken within a web forum.

The document is structured as follows: Chapter 2 includes information about ontologies: definition, types of ontology, uses of ontologies, etc. Chapter 3, is a state of the art related to the Semantic Web and the different semantic web languages used to represent knowledge by specifying ontologies and rules among their concepts. Later, Chapter 4 discusses aspects related to development of ontologies for the purpose of knowledge modeling and representation and the use of the most prominent ontologies in general and in the context of eLearning. Finally, the last Chapter takes all approaches seen so far one step further and develops an ontology-based framework as well as effective techniques for knowledge modeling and representation of the online sessions from collaborative learning performed within forums. Validation of such framework is also addressed in Chapter 5.

1.2 Methodology

The aim of this document is to provide a state-of-the-art overview on semantic web and its focus in eLearning, and to present a framework that deals with collaborative data from web forums. Firstly, an extensive literature research was conducted, searching in data bases and search engines for general terms (e.g., “ontologies”, “CSCL ontologies”, “eLearning Standards”, “knowledge representation”) to become a basic idea and common knowledge of the terms and definitions. In a second step, we then refined the search, using more specific terms. In general, our search terms remained broadly during the first steps to cover approaches in learning, conceptual modeling, knowledge representation and Computer Supported Collaborative Learning. The following data bases and search engines were used to find relevant literature:

- ISI Web of Knowledge
- ScienceDirect
- Google

Further resources were technical journals and special issues of these journals, books, and conference proceedings. Access to those media was either provided via the libraries of the Open University of Catalonia or the media were freely accessible via internet. Some of the books used to write this document are:

- Enabling Semantic Web Services (Fensel and others 2007),
- Ontology Management (Hepp, De Leenheer, De Moor 2007), and
- Ontological engineering (Gómez-Perez, Fernandez-López, Corcho 2004).

Moreover, references from relevant articles were checked for other studies and projects. We also considered information provided from special track papers, workshops, and working groups. For other projects of the European Community and other funding organizations relevant for this review, we searched through the respective websites.

Thereafter, the lessons learnt have been applied to create a framework that allows dealing with the collaborative information that occurs during the collaboration within learning experiences in virtual learning environments.

2 Theoretical foundations of ontologies

In this chapter, we define what an ontology is in the field of computer information systems. To do so, we compare Gruber's definition (Gruber 1993a), which is the most common, to other definitions. The comparison shows that the other definitions rewrite, interpret or specialize Gruber's definition. Thereafter, ontologies are classified according to their generality and expressivity, and the concepts of conceptual schema and ontology are compared to demonstrate their close relationship. The roles that an ontology may play in an information system are presented at the end of this chapter.

Section 2.1 presents different ontology definitions used in the field of computer information systems. Section 2.2 identifies the different elements that may comprise an ontology. A classification of the main ontologies is presented in Section 2.3, and the relation between ontologies and conceptual schemas is studied in Section 2.4. Finally, Section 2.5 reviews the relationship between ontologies and information systems, that is, how information systems use ontologies and for what purpose.

2.1 What is an Ontology?

The term *ontology* defines a philosophical discipline. In an informal and comprehensible way, we can define ontology as a branch of the philosophy that deals with the nature and organization of reality. Therefore, ontologies allow us to answer questions like *How many kinds of things exist? Animals, machines, stones, and so on; How are these things related? Animals and stones are tangibles;* and *In what way are they different? Animals are living beings, but stones are not.*

Even though the discipline was not defined until the seventeenth century, its practice dates from Ancient Greece. Today, ontologies are not only created by philosophers but also by computer scientists. Therefore, it is necessary to differentiate the meaning of the term *ontology* for each of these disciplines.

The Merriam-Webster dictionary[1] provides the following definitions:

1. *A science or study of being: specifically, a branch of metaphysics relating to the nature and relations of being; a particular system according to which problems of the nature of being are investigated.*
2. *A theory concerning the kinds of entities and specifically the kinds of abstract entities that are to be admitted to a language system.*

Philosophers tend to use the first definition. Computer scientists, however, use the second definition. This document focuses on the use of ontologies in computer science.

2.1.1 Ontologies and Computer Science

The use of the ontologies is relatively new in computer science and, as Welty said (Welty 2003), most of the members of scientific and engineering communities are using them without really knowing what they are. As a result, ontologies are constantly being redefined in an effort to eliminate the ambiguities of previous definitions. The goal of this section is to review the most prominent definitions. If we study these definitions we can conclude that, although there are many of them, most of them are equivalent.

The most used definition of an ontology in the field of computer science is Gruber's (Gruber 1993b). He defines (Calvanese, Lenzerini, Nardi 1998)an ontology as **the explicit specification of a**

[1] http://www.merriam-webster.com/

conceptualization, a conceptualization being an abstract and simplified view of the world we want to represent.

From the previous definition we can infer that an ontology should be written in a language. According to Gruber's definition, an ontology expressed in a natural language would be an ontology. However, this ontology would be useless in the field of computer science because no program is able to retrieve and use its knowledge. To solve that problem, ontology languages have been created in order to specify a domain in formal terms. Ontology languages tend to define their constructions using well-defined semantics in order to reduce the problem of ambiguity in their interpretation. The low expressivity of ontology languages (compared with any natural language) enables computers to retrieve and use the information of the ontologies. An example of an ontology language is *Description Logics* (Calvanese, Lenzerini, Nardi 1998), which provides a family of languages for representing knowledge with a precise semantics that limits the languages' expressivity to increase their capacity for inference.

According to Gruber's definition, the possibility of being interpreted by a program is not a necessary condition for an ontology. Nevertheless, it is a rule that all ontologies have satisfied so far. Gruber also explains that ontologies are designed for satisfying a set of requirements and that several design criteria should therefore be taken into account during their creation. Gruber argues that choosing one design criterion or another depends on the purpose of the ontology, but not on the nature or truth of its content.

As Gruber argued in an interview (Gruber 2004), the fact that ontologies are designed has several implications. When an ontology has been designed to satisfy functional goals (data interchange, unification, representation, communication, etc.) we do not need to worry about its truthfulness, its generality or whether it contains all the information on a given domain. The only thing we need to worry about is whether it fulfills the functions for which it was created.

Gruber's definition has been criticized by Guarino and Giaretta (Guarino and Giaretta 1995) for the definition of conceptualization it is based on, which he adopted from (Genesereth and Nilson 1987). According to Genesereth and Nilsson, a conceptualization is a set of extensional relationships that define the particular state of a given domain. This definition contradicts the intuitive definition of conceptualization, which is intensional, because it defines the different states of a given domain instead of a particular one.

Guarino and Giaretta also differentiate between the terms **Ontology** and **ontology**. To disambiguate the terms, they propose using the term *ontology* to define ontologies in the field of the computation and the term *conceptualization* to refer to the philosophical view of a domain (*Ontology*). They define conceptualization as a system of categories that represents a given view of the world or, in other words, the view of the world that a given person (or group of persons) has of a domain (or a set of domains).

The quality of Gruber's definition is improved by this redefinition of conceptualization because it differentiates the domain to be represented from the representation itself. This differentiation allows one to clearly see that two ontologies may be different, even when they represent the same domain, because they may be written in different languages, for different authors, with different goals in mind, etc. and the same domain Ontology can be obtained using different representation ontologies.

2.1.2 Other definitions of Ontology

2.1.2.1 Definitions of Ontology That Clarify Concepts

Gruber's is the most accepted definition of ontology. Other authors have tried to create new definitions in order to reduce the ambiguity of his definition; however, these new definitions have sometimes served to increase the ambiguity rather than reduce it.

Guarino and Giaretta, for instance, having observed that in the computer science community the term *ontology* was being used to designate different things, found it necessary to identify the different meanings of the term *ontology* as used by this community (Guarino and Giaretta 1995): a philosophical discipline, an informal conceptual system, a formal semantic account, a specification of a conceptualization, a representation of a conceptual system via a logical theory and characterized by specific formal properties or only by its specific purposes, the vocabulary used by a logical theory, and a (meta-level) specification of a logical theory. Moreover, in the same paper they propose a new definition of ontology:

> *An ontology is a logical theory which gives an explicit, partial account of a conceptualization.*

Two years later, Borst (Borst 1997) tried to refine Gruber's definition and stated that

> *An ontology is a formal specification of a shared conceptualization.*

The goal of this redefinition is to eradicate the ambiguity of the term *conceptualization* as used in Gruber's definition. However, such a redefinition is unnecessary when we use Guarino and Giaretta's definition of conceptualization. In this case, a conceptualization represents a domain and is general enough to be shared by any expert in the domain.

Sometime later, Studer et al. (Studer, Benjamins, Fensel 1998) merged the definitions of Gruber and Borst to create a new definition:

> *An ontology is a formal, explicit specification of a shared conceptualization. "Conceptualization" refers to an abstract model of some phenomenon in the world by identifying the relevant concepts of that phenomenon. "Explicit" means that the type of concepts used and the constraints on their use are explicitly defined. "Formal" refers to the fact that the ontology should be machine-readable. "Shared" reflects the notion that an ontology captures consensual knowledge, that is, knowledge that is not exclusive to one individual but accepted by a group.*

As in the previous case, this definition refines Gruber's definition, makes the concepts clear and reduces the ambiguities.

Even though many researchers have redefined the definition of the term *ontology* to create a more concise one, here we wish to refer only to Noy and McGuinness' definition, which is part of some of the most used guidelines on how to create an ontology from scratch (Noy and McGuinness 2001):

> *An ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and relations among them.*

The disadvantage of more concrete redefinitions is that these redefinitions tend to be confined to a particular branch of computer science and are therefore not applicable to computer science in general. We can see a clear example of this in the previous definition, which requires the ontology to be interpretable by a machine. This restriction is too limiting and discards some conceptualizations that could be considered ontologies according to Gruber's definition. For example, an ontology whose purpose is to support the designer in the specification of an information system does not need to be machine-interpretable; in fact, it can be written descriptively by hand on paper, because this paper may provide the designer with enough information to enable him or her to learn new information on the domain of interest and establish the system's requirements more precisely. Hence, this ontology and similar ontologies which can be used to support the creation of information systems cannot be considered ontologies according to Noy and McGuinness' definition.

2.1.2.2 Definitions of Ontology That Deal with the Goals, Content and Structure of Ontologies

Other definitions place a greater emphasis on the process used to define the ontology and its elements that the semantic concept of an ontology itself. An example is the following definition given by Bernaras et al. (Bernaras, Laresgoiti, Corera 1996):

> *An ontology provides the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base.*

The definition reflects the authors' approach to creating an ontology, which is part of the KACTUS project (Schreiber, Wielinga, Jansweijer 1995). This project allows an ontology to be created from the knowledge base of a program.

(Swartout and others 1996) propose another strategy for creating ontologies. Their methodology uses a linguistic ontology called SENSUS (Knight and Luk 1994) as the basis for deriving a final ontology. They define an ontology as follows:

> *An ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base.*

According to this definition, an ontology may be used to create different knowledge bases that share the same taxonomy. Creating new leave elements results in a more specific ontology. Since the ontologies share part of their taxonomy, it is easier to share information..

The term ontology is sometimes used to define taxonomies, such as UNSPSC,[2] RosettaNet,[3] OpenDirectory,[4] and so forth. The ontology community differentiates between ontologies that are taxonomies and ones that model domains more thoroughly, defining them as *lightweight* and *heavyweight* ontologies respectively (Lassila and McGuinness 2001). *Lightweight* ontologies contain concepts and properties that describe those concepts. On the other hand, *heavyweight* ontologies add axioms and constraints to lightweight ontologies, thus making the meaning of their terms clear. Section 2.3 discusses this classification in more detail.

Ontologies have been used for several purposes (natural language processing, knowledge management, management, e-commerce, information integration, the Semantic Web, etc.) and in different fields (artificial intelligence, databases, software engineering, etc.). (Uschold 1998) takes this multidisciplinarity into account in his definition of the term *ontology*, that is, he makes the concept understandable to people who are not part of the artificial intelligence community:

> *An ontology may take a variety of forms, but it will necessarily include a vocabulary of terms and some specification of their meaning. This includes definitions, and an indication of how concepts are interrelated, which collectively impose a structure on the domain and constrain the possible interpretations of terms.*

[2] http://www.unspsc.org/

[3] http://www.rosettanet.org/RosettaNet/

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

Domain models have long been used in the fields of database and software engineering. These models also take concepts, relationships, properties and constraints into account. However, some authors (usually in the field of artificial intelligence) argue that domain models include less semantic restrictions than heavyweight ontologies.

Unfortunately, after more than a decade of research on ontologies, the term *ontology* has still not been well defined. In fact, there is no consensus on the main properties of ontologies: what an ontology should be, what kind of information it should contain or how completely the ontology should describe the target domain. Some authors, for example, argue that ontologies should include instances, while others believe that ontologies are intensional and that the union of an ontology with its extension (instances) is a knowledge base. However, according to some of the philosophical definitions of ontologies, such as Immanuel Kant's (1724-1804):

> *Ontology deals with things in general, it abstracts from everything particular. It embraces all pure concepts of the understanding and all principles of the understanding of reason.*

We can conclude that ontologies contain general concepts but not concepts particular to a given situation. Therefore, following this philosophical assumption, we can conclude that instances should not be included in ontologies.

To complicate matters even further, some authors confuse the terms *epistemology*, *ontology*, *taxonomy*, *semantic* and *meaning*, even though these terms have related but different meanings: epistemology is the study of how ideas and concepts may exist, ontologies tell us the concepts and ideas that exist, taxonomy provides ways of classifying the elements of an ontology, and semantics provides ways of resolving the ambiguities that may result from an incomplete representation of the universe of discourse.

This section does not pretend to be an exhaustive exposition of all the definitions of ontology so far, but rather an overview of the most relevant ones (a more complete discussion about ontology definitions may be seen in (Gómez-Perez, Fernandez-López, Corcho 2004)). In this section we have shown that, although there are obvious differences between the definitions presented, all of them converge on the same ideas: **ontologies help to represent consensual knowledge in a generic way** and **the knowledge represented by the ontologies should be shared by several agents** (whether persons or programs)

2.2 Ontology Structure

Although the general belief might be that Gruber's definition was the first in the field, a prior definition by Neches, Fikes et al. (Neches and others 1991) defines an ontology in terms of its structure:

> *An ontology is the definition of the basic terms and relations comprising the vocabulary of a topic area as well as the rules for combining terms and relations to define extensions to the vocabulary.*

According to this definition, an ontology contains, on the one hand, the set of terms and relationship types that represent a domain and, on the other, all the information that may be inferred from it. In fact, according to the definitions discussed in the previous section, it seems that almost all the definitions expect an ontology with a similar structure.

In general, the principal components of an ontology are concepts, relationships and axioms. A **concept** is something we have created in our mind in order to generalize the properties a set of objects have in common. A concept has an extension and an intension. The extension is the set of all its possible instances, and the intension is the set of the common properties of all its instances.

Relationships relate objects to each other and describe their interactions or properties. Ontologies tend to have two different kinds of relationships:

– **Taxonomic relationships** are binary and enable concepts to be organized in a tree structure using generalization/specialization relationships. These relationships allow one to specify that a concept (child) is a subtype of another concept (parent); therefore, the extension of a child must be included in the extension of a parent. This relationship type is in fact an inclusion integrity constraint between concepts of parent and child. However, due to the important role of such relationships in ontologies, these relationships are defined explicitly as a relationship type.

– **Non-taxonomic relationships** are n-ary and relate concepts in a generic way, that is, without a predefined meaning. Their names are usually verbs that define the semantics of the relationships. They tend to be subjected to some integrity constraints (cardinality, transitivity, symmetry, etc.) that allow semantic interpretations to be restricted. Some languages predefine different non-taxonomic relationship types with their proper semantics, such as the OWL language (Bechhofer and others December, 2003), which predefines the relationships *imports, backwardCompatible With, incompatibleWith, priorVersion, sameAs, inverseOf*, etc.

Axioms are used to constrain the possible values of the instances of an ontology. An **instance** represents a particular concept of a real world, and it is represented for a tangible or intangible object generalized in such a concept. Some authors prefer to use the name *integrity constraint* rather than *axiom*.

In theory, an ontology should not contain instances because it represents the conceptualization of a given domain. In fact, the combination of an ontology with its instances is known as a knowledge base (Noy and McGuinness 2001). However, determining whether something is an instance or a concept poses a very difficult question, whose answer depends on the goal of the ontology. For example, in a very general ontology, *"Dog"* may be an instance of the concept *"Tame Animal"*. However, in another ontology used in a veterinary clinic, *"Dog"* may be a concept, and each dog that visits the clinic, such as *"Idefix"*, should be an instance of that concept.

In recent years, two ontology languages that include support for representing instances have been devised: DAML+OIL (Connolly and others March, 2001) and OWL (Bechhofer and others December, 2003). As a result, people are becoming more accustomed to including instances in ontologies, thus converting, in practical terms, the ontologies into knowledge bases.

2.3 Ontology Classification

Several classifications of ontologies have been defined up until now. This section presents some of the more well-known classifications of ontologies.

1. Guarino's classification (Guarino 1998) classifies ontologies according to their content or abstraction level.
2. Lassila and McGuinness' classification (Lassila and McGuinness 2001) classifies ontologies according to the expressivity of their language.
3. Poli's classification (Poli 2002) classifies ontologies according to their expressivity and generality.

2.3.1 According to Guarino

In Figure 2.1, ontologies are classified according to the information they contain.

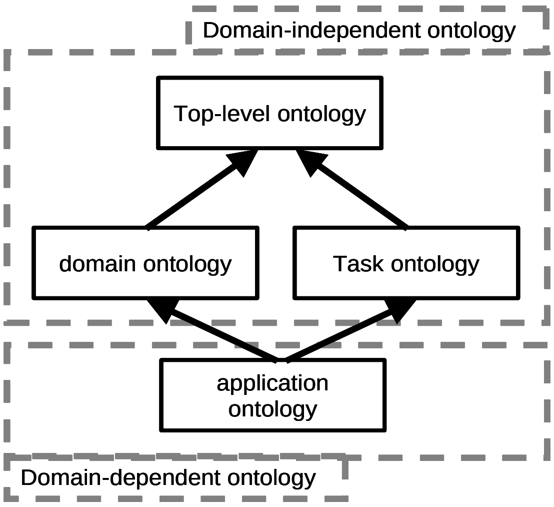


Figure 2.1. Classification of ontologies according to the level of abstraction (Conesa 2008)

- **Top-level ontologies** describe very general concepts, such as space, time, causal relationships and common sense knowledge. The knowledge defined in such ontologies is independent of any particular problem or domain, and these ontologies may therefore be used anywhere. Examples of these kinds of ontologies are SUMO,[5] Cyc ,[6] IFF,[7] Ontolingua,[8] GOL[9] and Sowa.[10]
- **Domain and task ontologies** describe the vocabulary related to general domains or tasks. They may be specified from a top-level ontology and may be reused within the same domain or task they deal with. Examples of domain ontologies are the medical ontology UMLS;[11] the linguistic ontologies Wordnet[12] and SENSUS;[13] and the genetic ontology GeneOntology[14] and the AKT[15] ontology, which try to model useful concepts for the reusability and maintainability of information. Examples of task ontologies are the *Business Process Management Ontology (BPMO)*[16] and the *Lifecycle integration of process plant data*[17] schema defined for the ISO.

---

[5] http://www.ontologyportal.org/
[6] http://www.cyc.com/
[7] http://suo.ieee.org/IFF/
[8] http://www.ksl.stanford.edu/software/ontolingua/
[9] http://www.ontology.uni-leipzig.de/Objectives.html
[10] http://www.jfsowa.com/ontology/toplevel.htm
[11] http://www.nlm.nih.gov/research/umls/
[12] http://www.cogsci.princeton.edu/%7Ewn/
[13] http://www.isi.edu/natural-language/projects/ONTOLOGIES.html
[14] http://www.geneontology.org/
[15] http://www.aktors.org/publications/ontology/
[16] http://www.bpiresearch.com/Resources/RE_OSSOnt/re_ossont.htm
[17] http://www.tc184-sc4.org/wg3ndocs/wg3n1328/lifecycle_integration_schema.html

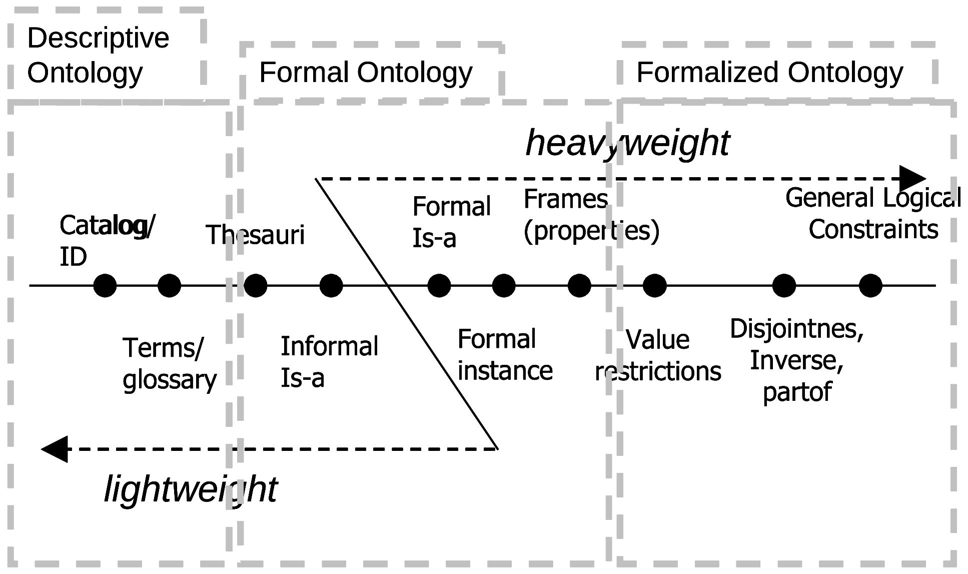


Figure 2.2. Classification of ontologies according to the level of formalism (Conesa 2008)

- **Application ontologies** describe the concepts dependent on a particular domain or task. They cannot be reused outside the particular domains they represent. A database schema, or ontologies created for use by just one program, may be considered application ontologies because they represent one or more specific domains or tasks.

2.3.2 According to Lassila and McGuinness

Ontologies are classified in Figure 2.2 according to the expressivity of the ontology language and how interpretable that language is for a computer.

- **Lightweight ontologies** are barely formal. Lassila and McGuinness identify the minimum elements of lightweight ontologies and add more elements to them. With each addition, a new kind of lightweight ontology is defined which is more formal than the previous ones. The simplest conceptualization that we can consider to be an ontology is a catalog. A catalog is a finite list of terms with an unambiguous interpretation.

  Glossaries are more complex than catalogs. Glossaries are defined using natural language as a list of terms and their meanings. Since natural language is ambiguous, these ontologies cannot be interpreted by a computer program. Thesauruses add more semantics to glossaries due to linguistic relationships (synonymy, heteronomy, etc.). In most cases the relationships may be unambiguously interpreted by a computer program.

  Some people do not believe that catalogs, glossaries and thesauruses are ontologies, because they argue that ontologies should incorporate the inheritance concept. Some ontologies use a general and informal notion of inheritance to structure their terms using generalization and specialization relationships. Ontologies use an informal notion of inheritance when the inclusion integrity constraints tied to the generalization and specialization relationships are not satisfied. The reason for such a violation may be that the ontology has been incorrectly constructed or that instances have been assigned incorrectly. In these kinds of ontologies we may find the category *Woman*, whose general category is *apparel* and which is specialized in *accessory* and *dresses*.

We can assign *Fragrance* as an instance of *Accessory*, but we cannot say that it is also an instance of *Woman* or *apparel*. Fortunately, these kinds of ontologies are rarely found today.

Lightweight ontologies are rarely considered to be ontologies because it is very difficult to create a program that is able to exploit the information they contain, due to their ambiguity and the difficulty of inferring new information using informal and unstructured knowledge.

– **Heavyweight ontologies** are more formal than lightweight ontologies. In contrast to lightweight ontologies, everybody considers heavyweight ontologies to be ontologies, even the less formal of them. Below, we discuss the different kinds of heavyweight ontologies and order them according to their degree of formalism.

  An ontology that contains formal inheritance (generalization/realization relationships) is considered a heavyweight ontology. An inheritance relationship (*IsA*) is formal when all its participants satisfy the inclusion integrity constraint tied to the relationship. Therefore, for all inheritance relationships it is true that:

$$\mathbf{IsA(}\mathit{Child,Parent}\mathbf{)} \rightarrow \forall i\ (\mathbf{InstanceOf(}\mathit{i,Child}\mathbf{)} \rightarrow \mathbf{InstanceOf(}\mathit{i,Parent}\mathbf{))}$$

  An ontology contains formal inheritance when all of its *IsA* relationships are formal.

  When there is a higher degree of formalism, we find that ontologies include classification relationships (InstanceOf). These relationships allow us to state that an individual is an instance of a concept.

  On another level we have ontologies with relationship types. In these ontologies a concept may include information on its properties.[18] Ontologies that allow the possible values of their properties to be constrained are more formal.

  Finally, the most formal ontologies contain general integrity constraints. These constraints allow the population of both entity types and relationship types to be restricted. These ontologies may also contain derivation rules, which are rules that infer the value of an element of the ontology using the information stored in the knowledge base.

  Finally, we can conclude that heavyweight ontologies are better than lightweight ones because they are more formal (they have axioms, generalization and specialization relationships, classification relationships, and general integrity constraints) and because they have constructions that are used to specify more and better information (such as attributes and instances). The increase in the number of constructions allows more information to be represented, and the increase in the level of formalism allows the meaning of the terms represented in the ontology to be made clear.

  The two classifications above are the most well known in the computer science field. Below, we present a more philosophical, generic classification that subsumes the previous ones. We believe that any ontology classification may be seen as a specialization or generalization of the following classification.

2.3.3 According to Poli

Depending on ontologies' degree of formalism and their content, they may be classified as any of the following.

[18] Note that an attribute is a particular case of a relationship type.

Depending on their degree of formalism:

1. Descriptive ontologies
2. Formal ontologies
3. Formalized ontologies

Depending on their content:

1. Domain-dependent ontologies
2. Domain-independent ontologies (also known as general ontologies)

According to Poli (Poli 2002), a **descriptive ontology** contains the information related to a domain (general or specific) that seems sufficient at a glance. On the other hand, a **formal ontology** distils, filters, codifies and organizes the content of a descriptive ontology in order to obtain a more formal one. These ontologies are deemed formal because they deal with pure categories that are not dependent on specific formalisms, such as thing, process, matter, everything, part of, etc. The formal codification, in the strict sense, corresponds to a third kind of ontology: a **formalized ontology**.

Depending on *what* the ontologies are representing, ontologies may be classified as 1) **domain-dependent ontologies**, which contain information that is dependent on a domain, or 2) **domain-independent ontologies**, which contain information that is general enough to be reused in any domain. These kinds of ontologies are also called **general ontologies**. A similar definition of general ontologies in the computer science field can be found in (Mizoguchi, Vanwelkenhuysen, Ikeda 1995) (Conesa, Palol, Olivé 2003). As we show in Figure 2.1, the classification of Guarino is a particular case of this classification.

Although there are more classifications of ontologies (Heijst, Spek, Wielinga 1997; Mizoguchi, Vanwelkenhuysen, Ikeda 1995), they are not explained here because they are either too similar to the ones presented or they can be considered specializations of the Poli classification.

Even though some authors try to create ontologies that are as formal as possible, other authors, such as Gruber (Gruber 2004), believe that there should be limits to the formalization of ontologies. The reason for this is quite simple: ontologies need to be machine-readable, but also human-readable. The last generation of ontology languages, such as OWL or DAML+OIL, allow more formal ontologies to be created, but this makes them harder for humans to read and understand than other conceptual modeling languages such as UML, ER or even logic.

2.4 Are the Terms Conceptual Schema and Ontology Equivalent?

As we have seen, an ontology may be considered an explicit representation of a conceptualization, in which the conceptualization represents an abstract view of the part of the world we want to represent. The database and knowledge engineering communities have been constructing domain models to represent conceptualizations for some time. These domain models are called conceptual schemas (CSs). Conceptual schemas take into account all the elements that are liable to be included in an ontology: concepts, properties, relationships, integrity constraints, inheritance relationships, instances, etc. Nevertheless, some authors argue that these schemas impose less semantic restrictions than ontologies do.

The previous argument may easily be refuted because most of the people in the artificial intelligence community use taxonomies as ontologies. In the general case, a CS will have more constraints than a taxonomy because it tends to include constructions that are not included in taxonomies, such as properties and integrity constraints (general, inclusion, covering, referential, etc.). Therefore, discarding CSs as ontologies would mean discarding most of the ontologies used today, which are generally taxonomies or thesauruses.

The differences between ontologies and conceptual schemas have been stated by several authors. However, their views on what a conceptual schema is are faulty because they regard it as the database schema of an information system rather than its conceptualization. (Gruber 1993a) states that, even though ontologies are similar to conceptual schemas, they have a different semantics. Gruber argues that conceptual schemas define relationships between data; on the contrary, ontologies define terms that represent knowledge. The conceptual schema of an information system is not its database schema, because it represents a conceptualization of the information system's domain. However, the database schema represents the information that the information system needs to store in order to perform its functions. If we take this distinction into account, it becomes clear that, even though the semantics of ontologies and databases schemas are different, this difference between conceptual schemas and ontologies does not exist if we follow the classic definition of conceptual schemas (Locopoulos 1992).

Other authors are reluctant to use conceptual modeling languages such as UML (OMG August, 2003) to represent ontologies. The quality of an ontology does not depend on the language of the ontology but on the information it represents and how this information meets its requirements. Therefore, we believe that conceptual modeling languages are as suitable as any other ontology language and in some cases even better, because there have better, more user-friendly tools for these kinds of languages.

We do not see significant differences between ontologies and conceptual schemas. Other authors have noted that there is a close relationship between conceptual schemas and ontologies (Karp 2000; Olivé 2004). In the following lines we try to define and justify the relationship between them.

All information systems (ISs) include information on one (or more) domains (Chandrasekaran, Josephson, Benjamins 1999; Sowa 2000). In the information systems field, that knowledge is usually called a conceptual schema (Olivé 2007). Conceptual schemas document users, analysts, and designers' common knowledge of a domain and the functions imposed on the information system (Locopoulos 1992); in another words, a conceptual schema represents the conceptualization of the domain of its IS. If we consider an ontology to be the explicit representation of a conceptualization, we can conclude that the explicit conceptual schema of an information system is an ontology that represents the domain the information system deals with.

Although most current ontologies only represent static information, we cannot discard ontologies that also represent dynamic knowledge. For example, Chisholm's ontology (Milton, Kazmierczak, Keen 2002) also allows some of a domain's dynamic knowledge to be modeled, particularly the changes in the state and the processes.

Conceptual schemas allow the static and dynamic knowledge of a domain to be represented. In fact, we can define a conceptual schema as (Olivé 2007) does:

Conceptual Schema = Domain Conceptual Schema + Functionality Specification

where *Domain Conceptual Schema* (DCS) represents the static part of an information system and the domain events, and the *Functionality Specification* (FS) the rest of the dynamic part.

Hence, we can conclude that:

> **The conceptual schema of an information system is the ontology that the information system needs to know in order to perform its functions (FS)**

The work (Olivé 2007) reinforces our conclusion because it compares the basic characteristics of one of the most used ontology languages, the *Web Ontology Language* or OWL (Bechhofer and others

December, 2003), to those of the de facto, standard language in conceptual modeling, UML. The paper concludes that both languages allow the knowledge base, which is the combination of the static conceptual schema and its information base (IB), to be modeled.

Up to now we have defined the relationship between conceptual schemas and ontologies, but this relationship is not symmetric. The relationship between ontologies and conceptual schemas is as follows:

> **The ontology[19] that represents the domain of an information system is equal to the structural conceptual schema of the information system (part of the DCS) plus one possible instantiation of it (IB).**

Due to the closeness of conceptual schemes and ontologies, all the design criteria applicable to an ontology (clarity, coherency, extensibility, codification independence and minimum ontological consensus (Gruber 1993a)) are also applicable to conceptual schemas. The same happens for conceptual schemas' design criteria (correctness, unambiguity, completeness, consistency, verifiability and modifiability; (Board 1998)), which are also applicable to ontologies.

Since we believe that ontologies and conceptual schemas are basically the same, the validity of an ontology may be measured by the same metrics than the validity of a conceptual schema. In fact, some of the recent research papers that define metrics to evaluate the quality of ontologies take advantage of measures used in the fields of software and conceptual schemas from long time ago, such as (Tartir and others 2005) that uses the cohesion measure to evaluate the quality of ontologies.

From a theoretical point of view, assessing the quality of an ontology (or its validity) is mainly validate whether the ontology satisfies the requirements for what it was created (Gruber 2004). When the ontology has been created to be used in the context of an information system then the validity of the ontology may be considered equivalent to the validity of the conceptual schema of the information system. Therefore, in such a case the validity of the ontology can be assessed by using the same measures or metrics used in assessing the validity of conceptual schemas: the ontology (CS) is valid when it contains all the knowledge necessary to satisfy the functions of the system (information system).

2.5 Ontologies and Information Systems

Since Gruber's definition, ontologies have become very popular. In the information systems field, ontologies have been used (and are still used) in knowledge engineering, information management, interoperability, conceptual modeling, the Semantic Web and integration. Ontologies are trusted because of their capacity for capturing the common and shared knowledge of a domain, which can be interpreted by both humans and programs.

Several authors have defended and studied the use of ontologies in the information systems field (Chandrasekaran, Josephson, Benjamins 1999; Pisanelli, Gangemi, Steve 2002; Smith ; Yair Wand and Weber 2004). Many information systems currently use ontologies to execute their tasks more efficiently.

As mentioned previously, every information system embeds its own conceptualization, either implicitly or explicitly (Olivé 2004). When an information system uses conceptualization explicitly, we can say that it is ontology-driven. When an ontology is used in the creation of an information system, we may speak of ***Ontology-driven Information System Development (ODISD)***; when an ontology is used in its execution, we may refer to the system as an ***Ontology-driven Information System (ODIS)***

[19] We assume that ontologies also contain instances because all current ontology languages allow instances to be represented.

(Guarino 1998). Below, we present several examples of the different ways in which an information system may use an ontology.

Ontology-driven Information System Development (ODISD). An ontology (or set of ontologies) may be used in the development of an information system for different purposes:

- To improve consensus between the different stakeholders involved in the creation of an information system, such as analysts, designers, programmers, clients, users, etc. (Holsapple and Joshi 2002). A method for creating ontologies using a cooperative approach may be defined, for example.
- To reuse its knowledge in the conceptual modeling activity. The ontology knowledge is written for domain experts, who are usually better acquainted with the domain than the analysts and designers of the information system. Examples of this are KRAFT (Pazzaglia and Embury 1998), which defines a bottom-up approach that extracts an ontology from a set of shared ontologies; (Maedche and Staab 2001) define a framework that allows ontologies to be obtained by refining a base ontology using e-learning techniques; Borst (Borst 1997) studies ontology reuse for sharing and reusing knowledge; (Kishore, Zhang, Ramesh 2004) define a methodology for creating ontologies and create domain frameworks from the ontologies created in order to validate them; (Ciancarini and Presutti 2002) use an ontology to create all the conceptual schemas needed for a website and to mark its web pages with semantic information; (Cristani and Cuel 2004) define a meta-methodology that allows different ontologies and methodologies to be used in the ontology creation process; and finally, (Falbo, Guizzardi, Duarte 2002) use knowledge of domain ontologies to derive the concepts and integrity constraints of a conceptual schema.
- To use its knowledge to learn about the domain of a given information system and the different tasks and functions in this domain. An example is given by (Sugumaran and Storey 2002), who describe an experiment in which several users use ontologies to create conceptual schemas and learn more about the domain of the information system to be developed.
- To use its knowledge to validate conceptual schemas created previously. The knowledge may be useful in detecting incoherence in the specification of an information system. Examples are provided by (Shanks, Tansley, Weber 2003), who study how the use of ontologies can improve the validation process in the conceptual modeling activity, and (Guarino and Welty 2002) , who explain how to use the ontology Ontoclean to validate other ontologies in its creation process.

Ontology-driven Information Systems (ODIS). In general, the use of ontologies in the execution of an information system improves its efficacy. In an information system, an ontology may be used for the following reasons:

- To improve communication between different agents (persons or programs), provide support in the communication language or facilitate consensus among different collectives. (Bagüés and others 2003) give an example in which an application for portable devices (PDAs) allows the health of elderly people to be monitored remotely and continuously. The system uses an operational ontology that allows the agents to communicate between themselves at a semantic level.
- To support the integration of different data sources, as in (Ras and Dardzinska 2004), who use a task ontology to solve semantic inconsistency problems when global queries are being translated into local queries.
- To establish interoperability between different applications. In (Embley 2004), for example, a domain ontology is used to establish dynamic interoperability between software agents.

- To support natural language interpretation. In (Mahesh and others 1996), knowledge of the Cyc ontology is applied to solve interpretation problems in natural language and automatic translation.
- Because it is the main component of the Semantic Web. Ontologies allow the semantic content of web pages to be modeled; they therefore enable certain semantic queries to be performed rather than textual ones. Examples are given by (Wollersheim and Rahayu 2002), in which a medical ontology is used to create a dynamic ontology that classifies the concepts of medical web pages, and (Guarino, Masolo, Vetere 1999), in which an ontology is used to increase the performance of queries related to telephone directory websites.

Obviously, the higher the quality of the ontology used, the greater the benefits of using it. However, creating high quality ontologies is a very hard task, although several ontology libraries are freely available on the Internet. These libraries contain ontologies that can be useful for a domain or a set of domains. Examples of free ontology libraries are the Ontololingua[20] and DAML[21] libraries. Other commercial ontologies (or libraries) are UNSPSC[22], RosettaNet[23] and Cyc[24].

2.6 Chapter summary

In this chapter we have broadly described and discussed what and ontology is and clarified concepts, goals, content and structure of ontologies starting with the many efforts to define it being the common view in computer information systems as the explicit specification of a conceptualization. Then, we have identified the main elements that may comprise an ontology, namely concepts, relationships and axioms. A great deal of perspectives is found around on the differences between ontologies and conceptual schemas being the consensuated conclusion that no significant distinguish them. We conclude the chapter with an overview and discussion on the use of ontologies in the information systems field and benefits achieved for the creation and execution of information systems.

[20] http://www.ksl.stanford.edu/software/ontolingua

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

[22] http://www.unspsc.org

[23] http://www.rosettanet.org

[24] http://www.cyc.com

3 The Semantic Web: Ontology languages

A major drawback of XML is that XML documents do not convey the meaning of the data contained in the document. Exchange of XML documents over the Web is only possible if the parties participating in the exchange agree beforehand on the exact syntactical format (expressed in XML Schema) of the data. The Semantic Web (Lee, Hendler, Lassila 2001) allows the representation and exchange of information in a meaningful way, facilitating automated processing of descriptions on the Web.

Annotations on the Semantic Web express links between information resources on the Web and connect information resources to formal terminologies– these connective structures are called ontologies(Fensel 2004) . Furthermore, ontologies facilitate interoperation between information resources through links to the same ontology or links between ontologies.

The general conviction held by the W3C is that the Semantic Web needs an ontology language that is compatible with current Web standards and is in fact layered on top of them. The language needs to be expressed in XML and, preferably, should be layered on top of RDF(S).

An often used depiction of the vision of Semantic Web languages is the“Semantic Web layer cake.” The original layer cake, which featured a rules language layered on top of the ontology language, was presented at XML2000 by Tim Berners-Lee, director of the World Wide Web Consortium (W3C). It turns out that rules languages cannot be layered on top of the Web Ontology Language OWL in a straightforward manner (Kifer and others 2010) ; this triggered a refinement of the layer cake, depicted in Fig. 3.1, where rules feature next to OWL, on top of a common layer.

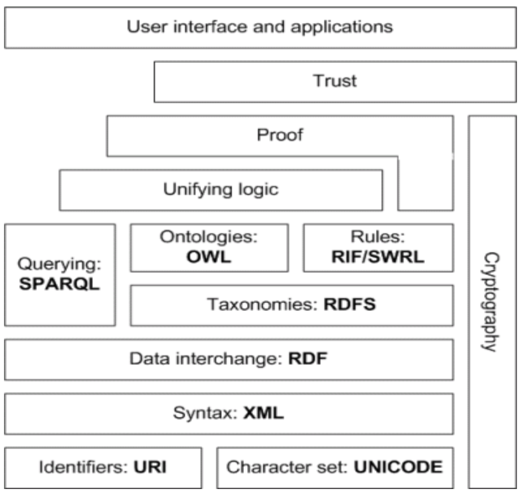


Fig. 3.1 The Semantic Web language layer cake (source from[25])

The bottom layers in the layer cake, i.e. Unicode and URI and XML (Schema), consist of existing Web standards and provide a syntactical basis for Semantic Web languages. Unicode provides an elementary character-encoding scheme, which is used by XML. The URI (uniform resource identifier) standard provides a means to uniquely identify and address documents and, more generally, resources on the Web. All concepts used in the languages located higher in the layer cake can be specified using Unicode and are uniquely identified by URIs.

[25] http://en.wikipedia.org/wiki/File:Semantic-web-stack.png

We shall describe the RDF(S), OWL, and rules layers below. We shall not cover the logic, proof, and trust layers here. Placing the logic layer on top of the OWL and rules layer is somewhat controversial, since OWL and rules languages are grounded in logic. Some argue that a more expressive logic language should be layered on top of the ontology language (Patel-Schneider and Fensel 2002). It could also be argued that this is not an appropriate layering; that is, that OWL and rules should be the top languages and that applications should use that layer directly. The proof and trust layers are not well-understood, but most likely refer to the application and not to any specific language. For instance, the application could prove some statement by using deductive reasoning, and a statement could be trusted if it had been proven and digitally signed by some trusted third party. The agent would very likely play an important role in the trust layer because it is the user that should decide whether or not an information source should be trusted.

In the remainder of this chapter we discuss the languages of the layer cake.

3.1 Ontology Languages

There are several languages that can be used to implement an ontology. Choosing the right language for each ontology is a key point. In the last decades, many ontology implementation languages have been created and other general Knowledge Representation (KR) languages and systems have been used for implementing ontologies though these were not specifically created with this purpose.

In the past, the selection of an ontology language is mostly based on the personal preferences of the ontology designer, instead of thinking about the KR and inference mechanisms needed by the application that uses the ontology. Nowadays, the XML-based languages are the most used to represent ontologies, being the Web Ontology Language the most used from them all.

In this section we will review the most prominent ontology representation languages.

3.1.1 Origins and evolution

At the beginning of the 1990s, a set of AI-based ontology languages was created. Basically, the KR paradigms underlying such ontology languages were based on first order logic (e.g., KIF), on frames combined with first order logic (e.g., CycL, Ontolingua, OCML and FLogic), and on description logics (e.g., LOOM). OKBC was also created as a protocol to access ontologies implemented in different languages with a frame-based KR paradigm.

CycL (Lenat and others 1990) was one of the first languages to be created. CycL is based on frames and first order logic and was used for building the Cyc Ontology.

At 1991 LOOM (MacGregor 1991) was built, though it was not intended to implement ontologies but for general knowledge bases. LOOM is based on description logics (DL) (Calvanese, Lenzerini, Nardi 1998) and production rules and provides automatic concept classification features. OCML (Motta 1999) was developed later, in 1993, as a kind of “operational Ontolingua”. In fact, most of the definitions that can be expressed in OCML are similar to the corresponding definitions in Ontolingua. OCML was built for developing executable ontologies and models in problem solving methods. Finally, in 1995 FLogic (Kifer, Lausen, Wu 1995) was developed as a language that combined frames and first order logic though it did not have Lisp-like syntax.

KIF (Genesereth, Fikes, Computer Science Department, Stanford University 1992) was created later, in 1992, and was designed as a knowledge interchange format; KIF is based on first order logic. Since ontologies were difficult to create directly in KIF, Ontolingua (Farquhar, Fikes, Rice 1997) was created on top of it. Ontolingua was considered a standard de facto by the ontology community in the 1990s.

In the spring of 1997, the High Performance Knowledge Base program (HPKB) was started. This research program was sponsored by the Defense Advanced Research Projects Agency and its objective was to solve many of the problems that usually appear when dealing with large knowledge bases (concerning efficiency, content creation, and integration of the content available in different systems). One of the results of this program was the development of the OKBC (Open Knowledge Base Connectivity) protocol. This protocol allows accessing knowledge bases stored in different Knowledge Representation Systems, which may be based on different KR paradigms. Of the languages aforementioned Ontolingua, LOOM and CycL are OKBC compliant.

The boom of the Internet led to the creation of ontology languages for exploiting the characteristics of the Web. Such languages are usually called web-based ontology languages or ontology markup languages. Their syntax is based on existing markup languages such as HTML (Raggett et al., 1999) and XML (Bray et al., 2000), whose purpose is not ontology development but data presentation and data exchange respectively.

The first ontology markup language to appear was SHOE (Luke and Heflin April, 2000). SHOE is a language that combines frames and rules. It was built as an extension of HTML, in 1996. It used tags different from those of the HTML specification, thus allowing the insertion of ontologies in HTML documents. Later its syntax was adapted to XML.

The rest of ontology markup languages presented here are based on XML. XOL (Karp, Chaudhri, Thomere 1999) was developed as a XMLization of a small subset of primitives from the OKBC protocol, called OKBC-Lite. RDF (Klyne, Carroll, McBride 2004) was developed by the W3C (the World Wide Web Consortium) as a semantic-network based language to describe Web resources. Its development started in 1997, and RDF was proposed as a W3C Recommendation in 1999. The RDF Schema (Brickley and Guha December, 2003) language was also built by the W3C as an extension to RDF with frame-based primitives. The combination of both RDF and RDF Schema is normally known as RDF(S).

These languages have established the foundations of the Semantic Web. In this context three more languages have been developed as extensions to RDF(S): OIL, DAML+OIL, and OWL. OIL (Fensel and others 2000) was developed at the beginning of the year 2000 in the framework of the European IST project On-To-Knowledge . It adds frame-based KR primitives to RDF(S) and its formal semantics is based on description logics. DAML+OIL (Connolly and others March, 2001) was created later by a joint committee from the US and the EU. DAML+OIL adds DL-based KR primitives to RDF(S). In 2001 the W3C formed a working group called Web-Ontology (WebOnt) Working Group 4 .The aim of this group was to make a new ontology markup language for the Semantic Web. The result of their work is the OWL language (W3C March, 2003). OWL covers most of the features of DAML+OIL and has renamed most of the primitives that appeared in that language.

3.1.2 Ontology Markup Languages

In this section, the following ontology markup languages are presented: SHOE, XOL, RDF(S), OIL, DAML+OIL, and OWL.

The syntax of these languages is based on existing Web markup languages like HTML and XML . We assume that the readers will be familiar with both languages.

3.1.2.1 SHOE

SHOE (Luke and Heflin April, 2000) was created as an extension of HTML with the aim of incorporating machine-readable semantic knowledge in Web documents. It provides specific tags for representing ontologies. As these tags are not defined in HTML, the information inside them is not shown in standard Web browsers. There is also a slight variant of the SHOE syntax for XML compatibility. In our examples, we have used HTML syntax.

The main objective of the SHOE language was to make it possible to collect meaningful information about Web pages and documents with the aim of improving search mechanisms on the Web. Consequently, its intended use can be summarized in the following three steps: (1) to define an ontology that describes concepts and relationships between them; (2) to annotate HTML pages with concept instances that describe such pages or other pages, and (3) to let agents search SHOE annotated Web pages to keep information updated and to allow retrieving semantic information.

SHOE developers have also contributed to the creation of other languages like DAML+OIL and OWL.

3.1.2.2 RDF and RDF Schema

RDF (Klyne, Carroll, McBride 2004) stands for Resource Description Framework. It is being developed by the W3C to create metadata for describing Web resources. RDF data model is equivalent to the semantic networks formalism and consists of three object types: resources, properties and statements.

The RDF data model does not have mechanisms for defining the relationships between properties and resources. This is the role of the RDF Vocabulary Description language (Brickley and Guha December, 2003), also known as RDF Schema or RDFS.

RDF(S) is the term commonly used to refer to the combination of RDF and RDFS. Thus, RDF(S) combines semantic networks with frames but it does not provide all the primitives that are usually found in frame-based knowledge representation systems. In fact, neither RDF, nor RDFS, and nor their combination in RDF(S) should be considered as ontology languages per se, but rather as general languages for describing metadata in the Web.

RDF(S) is widely used as a representation format in many tools and projects, and there exists a huge amount of resources for RDF(S) handling, such as browsing, editing, validating, querying, storing, etc. In the section about further readings, we provide several URLs where updated information about RDF(S) resources can be found.

RDF(S) provides the most basic primitives for ontology modeling, achieving a balance between expressiveness and reasoning. It has been developed as a stable core of primitives that can be easily extended. In fact, as we will discuss later, languages such as OIL, DAML+OIL, and OWL reuse and extend RDF(S) primitives.

The RDF data model is equivalent to the semantic network KR paradigm, as explained by Staab and colleagues (Staab and others 2007), and by Conen and Klapsing (Conen and Klapsing 2000). A semantic network is a directed labeled graph composed of a set of nodes and a set of unidirectional edges, and each has a name. Nodes represent concepts, instances of concepts and property values. Edges represent properties of concepts or relationships between concepts. The semantics of the network depends on the node and edge names. The semantic network KR paradigm has less expressiveness than the frame-based KR paradigm, since it does not allow representing, for instance, default values and cardinality constraints on attributes.

The RDF data model consists of three components:

- Resources, which are any type of data described by RDF. Resources are described with RDF expressions and are referred to as URIs (Uniform Resource Identifiers) plus optional anchor identifiers.
- Properties, which define attributes or relations used to describe a resource.

- Statements, which assign a value to a property in a specific resource. Just as an English sentence usually comprises a subject, a verb and objects, RDF statements consist of subjects, properties and objects. For instance, in the sentence “John bought a ticket”, John is the subject, bought is the verb, and ticket is the object. If we represent this sentence in RDF, John and ticket are resources, denoted graphically by nodes, while bought is a property, denoted graphically by an edge.

Not only can resources be the objects of a RDF statement, but RDF statements can also be objects themselves. For example, in the sentence “John said that Peter bought a ticket”, John is the subject, said is the property and Peter bought a ticket is the object, which can also be decomposed, as we did before. This is known as reification in RDF.

It is important to note that the RDF data model does not make any assumption about the structure of a document containing RDF information. That is, the statements can appear in any order in a RDF ontology.

Some comments have to be made on RDF(S) semantics. At present, some studies are being carried out by the W3C on the definition of the RDF(S) model theory (Hayes 2001). Previous to this work, the logical interpretation of this language (based on semantic networks) was explored by Conen and Klapsing (Conen and Klapsing 2000). The lack of formal semantics to create RDF(S) has caused several problems to define the semantics of other languages that extend RDF(S) like OIL, DAML+OIL, and OWL.

3.1.2.3 OIL

OIL (Fensel and others 2000) stands for Ontology Interchange Language and Ontology Inference Layer. It was developed in the context of the European IST project On-To-Knowledge. Like the other languages previously presented, for example, SHOE and RDF(S), OIL was built to express the semantics of Web resources. OIL was superseded by DAML+OIL.

OIL is a Web-based KR language that combines: (a) XML syntax; (b) modelling primitives from the frame-based KR paradigm, and (c) the formal semantics and reasoning support of description logics approaches. Thus OIL can be defined as a frame-based language that uses DL to give clear semantics and also to permit efficient implementations of reasoners for the language. According to the DL terminology, OIL is a SHIQ language.

OIL ontologies are not only implemented in XML, they can also be written as plain text files (known as OILs presentation syntax).

DL inference engines can be used for many purposes. They can be used to perform automatic classifications of ontology concepts in concept taxonomies, taking into account the subclass-of primitives and the slot constraints inside concept definitions. They can be used for constraint checking, taking into account the consistency of the concept taxonomies defined in the ontology.

3.1.2.4 DAML+OIL

DAML+OIL () was developed by a joint committee from the USA and the European Union (mainly OIL developers) in the context of the DARPA project DAML (DARPA Agent Markup Language). The main purpose of this language is to allow semantic markup of Web resources.

DAML+OIL has passed through several stages in its development. The last version of DAML+OIL was released in March 2001. Basically, this last version fixed some problems that were detected in the prior specification and changed some of the primitives of that version. None of these DAML+OIL versions used a layered structure for the language as OIL did.

DAML+OIL ontologies are written in XML (no plain text syntax, as in the case of OIL). And they can also be written with the triple notation for RDF. There are many tools, systems and applications to manage and use DAML+OIL ontologies. Many of them are been adapted to the OWL language, since this language supersedes DAML+OIL.

The use of DL classifiers permits performing automatic classifications of the ontology concepts, and detecting inconsistencies in this concept taxonomy. Independently of the inference engine that we use for reasoning with our DAML+OIL ontologies, multiple inheritance is allowed. Constraint checking can be performed on the values of properties and their cardinalities.

3.1.2.5 Web Ontology Language (OWL)

OWL (W3C March, 2003) is the result of the work of the W3C Web Ontology (WebOnt) Working Group, which was formed in November 2001. This language derives from and supersedes DAML+OIL. It covers most of DAML+OIL features and renames most of its primitives. As the previous languages, OWL is intended for publishing and sharing ontologies in the Web.

Like OIL, OWL is divided in layers: OWL Lite, OWL DL, and OWL Full. OWL Lite extends RDF(S) and gathers the most common features of OWL, so it is intended for users that only need to create class taxonomies and simple constraints. OWL DL, which stands for Description Logics, includes the complete OWL vocabulary, which is described in this section. Finally, OWL Full provides more flexibility to represent ontologies than OWL DL does.

There are 40 primitives in the OWL DL ontology (16 classes and 24 properties). Some RDF(S) primitives can be used in all the versions of OWL (OWL Lite and OWL DL), and that OWL Lite primitives can be used in OWL DL. OWL Full primitives are the same as the OWL DL ones. Like DAML+OIL, OWL is built upon RDF(S). Therefore, some RDF(S) primitives are reused by OWL, and OWL ontologies are written either in XML or with the triples notation for RDF.

As OWL is derived from DAML+OIL it shares many features with that language. The main differences between OWL and DAML+OIL are the following:

- OWL does not include qualified number restrictions (daml:hasClassQ,daml:cardinalityQ, daml:maxCardinalityQ, and daml:minCardinalityQ).
- OWL permits defining symmetric properties, which were not considered in DAML+OIL, with the primitive owl:SymmetricProperty.
- OWL does not rename the RDF(S) primitives reused by the language, as happened in DAML+OIL. For instance, rdfs:subClassOf, rdfs:subPropertyOf,etc.
- In OWL many DAML+OIL primitives have been renamed. For example, the primitive daml:toClass has been renamed as owl:allValuesFrom.
- OWL does not include the primitive daml:disjointUnionOf, since it can be effected by combining owl:unionOf and owl:disjointWith.

Due to its similarities with OIL and DAML+OIL, inference engines used for these languages (FaCT, RACER, TRIPLE, etc.) can be easily adapted for reasoning with it. As with other languages, these inference engines will permit performing automatic classifications of OWL ontology concepts, and detecting inconsistencies in OWL concept taxonomies.

Furthermore, we can say that multiple inheritance is allowed in OWL ontologies. In the semantics of OWL, however, there is no explanation on how conflicts in multiple inheritance can be solved. Constraint checking can be performed on the values of properties and their cardinalities.

OWL assumes monotonic reasoning, even if class definitions or property definitions are split up in different Web resources. This means that facts and entailments declared explicitly or obtained with inference engines can only be added, never deleted, and that new information cannot negate previous information.

As with DAML+OIL, many tools will be available for authoring OWL ontologies; tools capable of editing RDF(S) ontologies can also be used for developing OWL ontologies provided that the ontology developer uses the OWL KR primitives. In addition, RDF(S) query engines, storage systems, and parsers can be employed to manage OWL ontologies since they can be serialized in RDF(S).

3.2 Ontology Rule Languages

From the early days of the Semantic Web, rules have been seen as an important paradigm for representing and reasoning with knowledge on the Semantic Web. However, at the time of writing, activities have just gotten underway towards standardization of a rules language for the Semantic Web. With ontologies, one can express knowledge about classes, class hierarchies, properties, etc. Rules have a complementary expressiveness: with rules, one can express knowledge in the form “if A then B”. An example which is often used to motivate the use of rules is the “uncle” example, which says that “the brother of a person’s parent is that person’s uncle”:

$$\text{person}(?x) \wedge \text{parent}(?x, ?y) \wedge \text{brother}(?y, ?z) \Rightarrow \text{uncle}(?x, ?z).$$

3.2.1 Semantic Web Rule Language (SWRL)

SWRL (Horrocks and others 2004) is an extension of OWL DL which adds the expressive power of rules (without negation) to OWL; the above “uncle” example can be expressed in SWRL.

The basic SWRL constructs are Horn-like rules. However, whereas Horn rules have a conjunction of atomic formulas in the antecedent (body) of the rule and a single atomic formula in the consequent (head) of the rule, SWRL allows any OWL class description, property, or individual assertion in both the body and the head of the rule. In this way, SWRL diverges from traditional rules systems, which are based on Logic Programming or Deductive Databases.

Because SWRL combines the full expressive power of function-free Horn logic with an expressive description logic language, the key inferences tasks (e.g. satisfiability and entailment) are in general undecidable for SWRL.

3.2.2 F-Logic

F-Logic (Kifer, Lausen, Wu 1995), and, more specifically, the Horn subset of F-Logic extended with negation, has been proposed as an ontology and rule language for the Semantic Web (Kifer 2005). Rules in F-Logic are similar to Horn rules, with the distinction that besides atomic formulas, F-Logic rules also allow molecules in place of atomic formulas. Note that although the syntax of F-Logic seems higher-order, the language remains semantically in the first-order framework.

There are various kinds of molecules. An is-a assertion C:D states that C is an instance of the class D; a subclass assertion C::D states that C is a subclass of D. Data molecules of the form C[D ->> E] have the meaning that the attribute D of the individual C has the value E. Signature molecules of the form C[D =>> E] indicate that the class C has an attribute D and that all values associated with this attribute are of type E.

An important concept in F-Logic is object identity(Khoshafian and Copeland 1986). Each object (e.g. a class, instance, or method) has a unique object identifier, where an object identifier is in fact a term. In F-Logic, classes and methods are interpreted intentionally, which means that class identifiers and method identifiers are interpreted by themselves and not directly as sets or as binary relations, as is the case with concepts and roles in description logics. Classes and methods are first interpreted as objects in the domain, and these objects are then related to sets of objects and sets of binary tuples, respectively.

In F-Logic there is no distinction between classes and instances.An object identifier can denote a class, an instance, or an attribute, but there is no separation in the signature Σ of the identifiers denoting any of these. The advantage of such an overloaded concept of an object is that objects denote classes, instances, and attributes depending on the syntactic context, thereby allowing certain kinds of metastatements. For example, we might define an attribute parent, which is not related to the class parent:

person[parent =>> person].

Both SWRL and F-Logic have been proposed as rules languages for the Semantic Web. The main difference between the two proposals is that in SWRL, the rules language is seen as an extension of the ontology language OWL DL, whereas in the F-Logic (programming) proposal, ontologies aremodeled using rules.

3.3 Chapter summary

In this chapter we have presented the Semantic Web as the most common view of ontology for the Web as the Semantic Web connect Web information resources to formal terminologies, seeing these connective structures as ontologies. From the general conviction that Semantic Web needs an ontology language compatible with current Web standards and layered on top of Web standards many ontology languages for representing and reasoning with knowledge on the Semantic Web have appeared, being the most well-known examples based on markup languages, such as OWL and RDF. Ontology rule languages are also presented to meet the need for rules when expressing knowledge in the form of "if A then B" ad for deduction and transformation tasks. Main representation of ontology language is the extension of OWL, called SWRL.

4 Development of ontologies for knowledge modelling and representation

Information is data that makes a difference. It is intended to shape the person who receives it, to make some difference in his or her outlook or insight. It has shape and purpose (Davenport and Prusak 2000). Knowledge, on the other hand, is experiences, values, insights, and context, in addition to information. It develops over time, through the experience that includes what is absorbed through formal and informal learning. It is not a rigid structure and can deal with complexity in a complex way. Knowledge can judge new situations and information in light of what is already known, and judges and refines itself in response to new situations and information (Davenport and Prusak 2000).

For knowledge to be reused, it needs to be externalized or extracted, translated, transferred, adapted, and applied. When a team performs a task and its outcome is achieved, the team must then explore the relationship between action and outcome to gain common knowledge. This knowledge must be translated into a form usable by others and then transferred to others. Receivers of the knowledge will then adapt it for their own use and go on to perform other tasks (Dixon 2000).

Knowledge is one of the most important assets of organizations (Grant 1996). Organizational capabilities for the development, storage, sharing and effective use of knowledge have been particularly recognized as a strategic differentiator among competing firms/units (Nidumolu, Subramani, Aldrich 2001). Today's business environment is marked by high turnover and global competition. In this context, whereas good knowledge processes are important for many organizations, system solution providers are particularly under pressure to search for ways to gain a strategic edge over their competition by increasing productivity, quality and cutting costs of software development. In addition, organizations are addressing problems of delay in the completion of software development projects and budget overruns. In software development where intellectual capital is highly critical to product development success, one of the keys is to effectively reuse the knowledge acquired within and across projects (Lindvall and Rus 2002).

A common understanding of the knowledge is needed to be able to translate it and transfer it among different stakeholders, who have different contexts and thus different ways of understanding the problem for which the knowledge is needed. This common understanding must be placed in a repository where all stockholders can access it, adapt it to perform new tasks and then contribute new knowledge to the point where new stakeholders will use the new knowledge to perform even more tasks.

Repositories of knowledge about the real world can be found in the form of ontologies, which are being developed at two levels (Guarino 1998): 1) individual domain ontologies that capture concepts about a particular application domain, and 2) upper level ontologies that contain massive amounts of knowledge about the real world and are domain independent. The most well-known upper level ontology is Cyc (encyclopedia) project (Lenat and others 1990), which is an ambitious attempt to capture common sense knowledge about the world and encode it in a knowledge base. ResearchCyc, a version of Cyc has been made available for use by the scientific community.

In this chapter, an overview on construction, evaluation and use of ontologies in general is provided. This includes the support during the development of ontologies, namely tools and external resources, such as libraries and existing general large ontologies to be reused by domain ontologies..

4.1 Ontology Development

The motivation for ontology development includes: sharing a common understanding of the structure of information among people or software agents; enabling reuse of domain knowledge, making domain assumptions explicit; separating domain knowledge from operational knowledge; and analyzing domain knowledge (Noy and McGuinness 2001).

Most ontologies are developed manually with ontology developers satisfying functional goals (e.g., data interchange, unification, representation, communication). Ontology development is indeed difficult. The developer uses personal knowledge as well as knowledge acquired from accessible sources to identify key terms that are coded into classes or subclasses, and to establish relationships among them. Thus, it is difficult to assess an ontology's truthfulness, generality, or whether it contains all the information for a given domain. The only important criteria is often whether an ontology fulfills the functions for which it was created (Gruber 2004).

The potential benefits of realizing the Semantic Web through ontologies and semantic applications has led to the development of semi-automatic methodologies to create ontologies, including Text-to-Onto (Volz and others 2003), TANGO (Tijerino and others 2005), SENSUS (Swartout and others 1996), Knowledge bus (Peterson, Andersen, Engel 1998), Materialized Ontology View Extraction (MOVE) (Wouters and others 2004), Ontology Pruning and Refactoring (OPR) (Conesa 2008) and APOSDLE[26] tools.

In semi-automatic ontology development, specifically designed tools are used for tasks, such as gathering significant terms, or identifying appropriate relationships among them. Fully automated ontology development is difficult because solving non-objective design criteria may require designer intervention. Examples include determining whether a concept is a class or a data type or determining the covering and disjointness constraints of a set of concepts. Ontology development is, thus, labor-intensive and expensive because of the broad range of skills and knowledge required (Good and others 2006) . Moreover, the scale and complexity of ontologies are growing quickly with the rapid development of the Semantic Web (Li and others 2005).

4.1.1 Environments supporting ontology development

The best known ontology development and manipulation system is Protégé[27].It can facilitate rapid prototyping and application development and provides interoperability features (export function over RDF (Klyne, Carroll, McBride 2004), OWL (Bechhofer and others December, 2003), and XML (Bray and others February 2004)).

TopBraid Composer[28] provides full support for developing, managing, and testing knowledge models. It includes SPARQL queries, UML-like diagrams, geospatial mapping, visual RDF graphs, and ontology-driven forms. Composer provides multi-user support, scalable database back-ends, and can be used for editing RDFS/OWL files using various formats.

SWOOP[29] is a hypermedia-based OWL ontology creator, editor, and debugger. Its main advantage resides in its simplicity and ease of use for its user interface that is similar to a web-browser. SWOOP

[26] http://www.aposdle.tugraz.at/

[27] "The Protegé Ontology Editor and Knowledge Acquisition System." from http://protege.stanford.edu

[28] http://www.Topbraid.composer.com/index.html

[29] "SWOOP - Hypermedia-based OWL Ontology Browser and Editor." 2007, from http://www.mindswap.org/2004/SWOOP

is an open source project. The IBM IODT[30] (IBM Integrated Ontology Development Toolkit) supports ontology building, management, and visualization, and provides an OWL ontology repository, Minerva.

These tools have been effective in representing the concepts in domain ontologies. However, their major function is as a representation tool. The user is still responsible for identifying the terms, relationships, and constraints of a domain. This is however a challenge that should be addressed in an automated way. Embley (Embley 2004) argues that it is both possible and feasible to automate ontology development. To this end, methodologies must be developed for extracting and organizing knowledge such as, concepts from relational tables on the Internet. Other proposals involve searching and crawling web sites to extract the relevant information about a domain.

4.1.1.1 Ontology libraries

Research on information systems, Semantic Web, databases, artificial intelligence, and information integration has for all of them highlighted the need for libraries of common ontologies. The existence of ontology libraries would facilitate the identification and retrieval of ontologies that would fit the needs of a specific domain and application (Noy 2004) . Despite the perceived benefit there is a scarcity of libraries. Existing tools and development environments lack maturity, completeness, and efficiency. Attempts to optimize the use of existing ontologies focus on developing efficient semantic integration mechanisms, mostly based on mapping techniques to find a means “to relate the different ontologies to each other” (Obrst and others 2006).

Practical ontologies tend to be semiformal (Gruber 2004) and usually not well documented. There are inconsistencies in the availability of graphical presentations and meta-level information. It is important to know whether the links of ontologies point to pdf and html files (documents) or to OWL and RDF files. An ontology may not be very usable because one must identify all the concepts, relationships and constraints using an ontology language such as OWL or DAML .

4.1.1.2 Preexistent Large ontologies

Various large knowledge repositories have been created to support agents (humans or programs) to increase the intelligence of their tasks. Examples include Wordnet, Cyc, ConceptNet[31], Open Biomedical Ontologies[32], UMLS (Bodenreider 2004) and eClassOWL[33]. The Cyc ontology is a huge semantic repository that captures and represents common sense, with the quantity and quality of information it contains considered superior to other knowledge sources. Although the Cyc ontology is probably the most well-known repository of common sense knowledge, it is usually ignored by the Semantic Web community, possibly due to its usability problems (Conesa, Storey, Sugumaran 2010). ResearchCyc, however, has been used to support web queries through knowledge expansion and refinement.

4.1.1.3 Ontology Infrastructures

While repositories aim at being able to efficiently store and retrieve large collections of data (i.e. managing explicit facts), reasoners focus on deduction procedures to derive implicit knowledge.

Thus independent reasoner and repository realisations can be normally integrated via defined interfaces. However for efficient large-scale ontology support, repository and reasoner realizations have often some of the other functionality. For example ontology repository realisations provide

[30] http://www.alphaworks.ibm.com/tech/semanticstk

[31] http://web.media.mit.edu/~hugo/conceptnet/
[32] http://www.obofoundry.org/
[33] http://www.heppnetz.de/projects/eclassowl/

database-like functionalities with (typically limited) inferencing support. In turn, many reasoner realisations rely on an integrated repository.

In the following, we briefly describe existing reasoners and the supported ontology languages, their reasoning approaches, availability and interfaces. The overview is partially based on the Description Logic Reasoner site (http://www.cs.man.ac.uk/~sattler/reasoners .html).

- Cerebra Engine is a commercially developed C++-based reasoner. It implements a tableau-based decision procedure for general TBoxes (subsumption, satisfiability, classification) and ABoxes (retrieval, tree-conjunctive query answering using an XQuery-like syntax). It supports the OWL-API and comes with numerous other features.
- FaCT++ is a free open-source C++-based reasoner for SHOIQ with simple data types (i.e., for OWL-DL with qualifying cardinality restrictions). It implements a tableau-based decision procedure for general TBoxes (subsumption, satisfiability, classification) and incomplete support of ABoxes (retrieval). It supports the Lisp-API and the DIG-API.
- KAON2 (Motik and Sattler, 2006) is a free (free for non-commercial usage) Java reasoner for SHIQ 4 extended with the DL-safe fragment of SWRL. It implements a resolution-based decision procedure for general TBoxes (subsumption, satisfiability, classification) and ABoxes (retrieval, conjunctive query answering). It comes with its own, Java-based interface, and supports the DIG-API.
- OntoBroker is a commercial Java based main-memory deductive database engine and query interface. It processes F-Logic ontologies and provides a number of additional features such as integration of relational databases and various built-ins. The new version of OntoBroker offers the KAON2 API.
- Pellet (Sirin et al., 2007) is a free open-source Java-based reasoner for SROIQ 5 with simple data types (i.e., for OWL 1.1). It implements a tableau-based decision procedure for general TBoxes (subsumption, satisfiability, classification) and ABoxes (retrieval, conjunctive query answering). It supports the OWL-API, the DIG-API, and Jena interface and comes with numerous other features.
- RacerPro is a commercial (free trials and research licenses are available) lisp-based reasoner for SHIQ with simple data types (i.e., for OWL-DL with qualified number restrictions, but without nominals). It implements a tableau-based decision procedure for general TBoxes (subsumption, satisfiability, classification) and ABoxes (retrieval, nRQL query answering). It supports the OWL-API and the DIG-API and comes with numerous other features.
- OWLIM is semantic repository and reasoner, packaged as a Storage and Inference Layer (SAIL) for the Sesame RDF database. OWLIM uses the TRREE engine to perform RDFS, and OWL DLP reasoning. It performs forward-chaining of entailment rules on top of RDF graphs and employs a reasoning strategy, which can be described as total materialization. OWLIM offers configurable reasoning support and performance. In the "standard" version of OWLIM (referred to as SwiftOWLIM) reasoning and query evaluation are performed in-memory, while a reliable persistence strategy assures data preservation, consistency and integrity.

In the following we additionally discuss two implementations of ontology repositories: Jena and Sesame are the two most popular implementations of RDF stores. They play a separate role, as their primary data model is that of RDF. However, they deserve discussion, as they offer some OWL functionalities and limited reasoning support.

- Sesame (http://openrdf.org, Broekstra et al., 2002) is an open source repository for storing and querying RDF and RDFS information. OWL ontologies are simply treated on the level of

RDF graphs. Sesame enables the connection to DBMS (currently MySQL, PostgreSQL and Oracle) through the SAIL (the Storage and Inference Layer) module, and also offers a very efficient direct to disk Sail called Native Sail. Sesame provides RDFS inferencing and allows querying through SeRQL, RQL, RDQL and SPARQL. Via the SAIL it is also possible to extend the inferencing capabilities of the system. (In fact, this is how the OWLIM reasoner is realized.) The main ways to communicate with the Sesame modules are through the Sesame API or through the Sesame Server, running within a Java Servlet Container.

- Jena is a Java framework for building Semantic Web applications (http://jena.sf.net). It offers the Jena/db module which is the implementation of the Jena model interface along with the use of a database for storing/retrieving RDF data. Jena uses existing relational databases for persistent storage of RDF data; Jena supports MySQL, Oracle and PostgreSQL. The query languages offered are RDQL and SPARQL. Just as in Sesame, the OWL support is realized by treating OWL ontologies as RDF graphs. However, in addition Jena also provides a separate OWL API and allows integration with external reasoners, such as Pellet.

4.1.2 Ontology Development Tools

A clear focus of current ontology management tools is to support the development of ontologies with a wide range of editing features. The following description of ontology tools is not meant to be complete. Instead we chose tools which represent different philosophies due to their history, their target users, etc.

Starting with Protégé as probably the most popular ontology development tool we describe an environment with a long history and a large number of features which go beyond pure editing of ontology-files. Other environments supporting a range of tasks in the broader context of ontology development include the commercial tools such as TopBraid Composer™. We then present tools that focus on certain aspects, such as providing a native OWL editor reflecting its characteristics as Semantic Web language (SWOOP), offering a rich graphical interface (Altova Semantic Works™) or rule-support and semantic integration (OntoStudio®).

While most of the tools focus on RDF(S) and/or OWL as ontology language, two of the presented environments support other languages. Protégé as a hybrid tool supports its own frame-based representation as well as OWL and RDF(S). The frame-based format, which from a historical point of view is the “original” native representation of Protégé, is related to formats used in expert system shells. OntoStudio® offers a couple of functionalities based on the F-Logic language, which mainly concerns the creation and management of rules. The latter functionalities differ from the rule-features some of the other tools offer, since those support SWRL rules as an extension to OWL ontologies.

The following sections focus on the characteristics of the tools from a user’s perspective. The last sections provide a comparison of the core features and characterize the current state of ontology development tools.

4.1.2.1 Protégé

Protégé 3.2 (Gennari et al., 2002) is the latest version of the Protégé OWL editor (Knublauch et al., 2004), created by the Stanford Medical Informatics group at Stanford University. Protégé is a Java-based open source standalone application to be installed and run a local computer. It enables users to load and save OWL and RDF ontologies, edit and visualize classes, properties and SWRL rules (Horrocks et al., 2004), define logical class characteristics as OWL expressions and edit OWL individuals.

With respect to the supported languages Protégé is a hybrid tool. The internal storage format of Protégé is frame-based. Therefore Protégé has native frame-support. The support for OWL is provided by a special plug-in that fits into the Protégé plugin architecture. Another example of plugin is the versioning support in Protégé (Noy et al., 2004).

The Protégé-OWL API is built on top of the frame-based persistence API using “frame-stores.” The API provides classes and methods to load and save OWL files, to query and manipulate OWL data models, and to perform reasoning based on Description Logic engines. The API is designed to be used in two contexts: (1) development of components that are executed inside the Protégé UI, and (2) development of stand-alone applications (e.g. Swing applications, Servlets or Eclipse plugins).

The OWL APIs implementation rely both on the frame-based knowledge base for low level (file or DBMS based) triple storage, and both on the Jena APIs for various services, such as OWL parsing and data type handling.

The Protégé-OWL API can be used to generate a Jena Model at any time in order to query the OWL model, for example by means of the SPARQL RDF query language (Prud’hommeaux et al., 2007). Reasoning can be performed by means of an API which employs an external DIG compliant reasoner, such as RACER, FaCT++, Pellet or KAON2.

Protégé offers a proprietary framework for plugins enabling users to extend the tool. The possible plugins include custom widgets as well as additional storage backends. In contrast to platforms like Eclipse there is a predefined set of possible extensions, which excludes “plugins of plugins.”

Protégé has gained much popularity over the years and has a large user-base. Consequently a large number of plugins is available. The standard distribution contains plugins for graph-based visualization, import of different formats and many more. Additional plugins offer for example ontology merging functionalities. Apart from the community support through the Protégé website and mailing lists, there are Protégé regular user conferences.

For historical reasons Protégé has not been designed as a native OWL tool. As previously mentioned the OWL support is built on top of the frame-based storage API, but it also uses partly the Jena API for certain tasks. Protégé builds on a bridge between its internal triple store and the Jena API.

While Protégé offers a unique look and feel for both, frame-based ontologies and OWL ontologies, the implementation of an OWL API on top of a frame-based API has significant disadvantages over the design of a native OWL API. Consequently the next generation of Protégé OWL, which by the time of writing this text was only available as a prototype, is a standalone tool using a “pure” OWL API.

4.1.2.2 Altova SemanticWorks

SemanticWorks™ is a commercial OWL editor offered by Altova[34]. The most outstanding feature of the tool is the graphical interface. SemanticWorks™ supports the visual editing of OWL and RDF(S) files using a rich, graph-based multi-document user interface. The latter supports various graphical elements including connections and compartments.

The visualization of ontologies utilizes very similar mechanisms from the other Altova products, which are XML-based. This means they are syntax-oriented. There is for example hardly any difference between the visualization of meta-objects of OWL like owl:Class and a user class. This makes it difficult to get an overview on the user content of an ontology. Ontologies can be saved as .rdf, .rdfs, or .owl files and can be exported in their RDF/XML and N-Triples formats.

[34] http://www.altova.com/download/semanticworks/semantic_web_rdf_owl_editor.html

SemanticWorks™ does — in contrast to other tools presented in this section — not include direct interactions with reasoners for consistency checking, debugging, query processing etc. Thus the tool might be seen as a pure editor, rather than a development tool, especially when compared to tools like SWOOP. The latter also focuses on the creation and management of OWL-files, but includes for example debugging capabilities. The strength of SemanticWorks™ is the graphical interface with its navigation capabilities (e.g. dynamic expansion of elements with automatic layout).

4.1.2.3 TopBraid Composer

TopBraid Composer™ is a modelling tool for the creation and maintenance of ontologies[35]. It is a complete editor for RDF(S) and OWL models. TopBraid Composer™ is built upon the Eclipse platform and uses Jena as its underlying API. The following list contains some of the characteristics of the tool. It is implemented as an IDE-application using the Eclipse platform with all its advantages (such as the plugin concept).

TopBraid Composer™ supports consistency checks and other reasoning tasks. The system has the open-source DL reasoner Pellet built-in as its default inference engine, but other classifiers can be accessed via the DIG interface.

Historically the development of TopBraid Composer™ has its roots in Protégé OWL 10 . Thus some of the concepts of TopBraid™ are similar to those of Protégé, such as the generation of schema-based forms for data acquisition. The most obvious difference from a technical perspective is the usage of the Eclise platform as a base and the lack of the frame-based part.

The latter allows TopBraid Composer™ to build on an OWL/RDF(S) based infrastructure, but excludes the support for frame-based technologies. TopBraid Composer™ offers functionalities going beyond the creation and management of OWL/RDF(S) files. This includes the import of databases, XML-Schemas, UML and spreadsheets as well as a basic support for rules. The system supports rules in either the Jena Rules format or SWRL. Both types of rules are executed with the internal Jena Rules engine to infer additional relationships among resources. Rules can be edited with support of auto-completion and syntax checking.

Other features of TopBraid Composer™ include the visualization of relationships in RDFS/OWL resources in a graphical format and the support for the concurrent editing of several ontologies. TopBraid Composer™ provides an explanation feature for OWL DL that is based on Pellet — similar to SWOOP.

TopBraid Composer™ represents a complex ontology development tool suitable for a number of tasks that go beyond the creation of OWL/RDF(S) files. As the other Eclipse-based implementations, TopBraid Composer™ is extensible by custom plugins. TopBraid Composer™ does — in contrast to the historically related Protégé — mainly (if not only) target professional users rather than a large community.

4.1.2.4 Integrated Ontology Development Toolkit (IODT)

The Integrated Ontology Development Toolkit[36] (IODT) was developed by IBM. This toolkit includes the Ontology Definition Metamodel (EODM), EODM workbench, and an OWL Ontology Repository (named Minerva). EODM is derived from the OMG's Ontology Definition Metamodel (ODM) and implemented in Eclipse Modelling Framework (EMF). In order to facilitate software development and

[35] http://www.topquadrant.com/products/TB_Composer.html

[36] http://www.icewalkers.com/Linux/Software/522730/IBM-Integrated-Ontology-Development-Toolkit.html

execution, EODM includes RDFS/OWL parsing and serialization, reasoning, and transformation between RDFS/OWL and EMF-based formats. These functions can be invoked from the EODM Workbench or Minerva.

Minerva is an OWL ontology storage, inference, and query system based on RDBMS (Relational Database Management Systems). It supports DLP (Description Logic Program), a subset of OWL DL.

The EODM Workbench (see a screenshot in the following figure) is an Eclipse-based editor for users to create, view and generate OWL ontologies. It has UML-like graphic notions to represent OWL class, restriction and property etc. EODM Workbench built by using EODM, EMF, Graphic Editing Framework (GEF), which provides the foundation for the graphic view of OWL. It also provides two hierarchical views for both OWL class/restriction and OWL object/datatype property.

As an Eclipse-based Tool the EODM workbench benefits from all advantages of the Eclipse platform (coupling with other plugins, etc.). In addition to traditional tree-based ontology visualization, EODM workbench provides UML-like graphic notion. Class, DatatypeProperty and ObjectProperty in OWL share the similar notion as Class, Attribute and Association in UML. Detailed properties of OWL constructs are shown in the Property view.

The EODM workbench supports multiple views for ontologies, enabling users to visually split large models. These views are independent from each other but synchronized automatically.

Being based on Eclipse, EODB is extensible, similar to products like TopBraid Composer™ and OntoStudio®. It does however not offer the direct interaction with an underlying reasoner in the form that the latter tools to and therefore lacks comfortable consistency checks or testing features.

EODM is deployed and installed as a set of Eclipse plugins. It therefore does not offer the easy-to-use installation routines of the other environments, which are deployed as standalone tools.

Offering an EMF-based implementation of an OWL and an RDF(S) metamodel, EODM offers interesting opportunities for developers, such as the combination with other EMF-based technologies or the extension of the metamodel itself.

4.1.2.5 SWOOP

SWOOP (Kalyanpur et al., 2005) is an open-source hypermedia-based OWL ontology editor. The user interface design of SWOOP follows a browser paradigm, including the typical navigation features like history buttons. Offering an environment with a look and feel known from Web browsers, the developers of swoop aimed at a concept that average users are expected to accept within short time. Thus users are enabled to view and edit OWL-ontologies in a “Web-like” manner, which concerns the navigation via hyperlinks but also annotation features. SWOOP therefore provides an alternative to Web-based ontology tools but offers additional features such as a plugin-mechanism.

SWOOP is designed as a native OWL-editor, which supports multiple OWL ontologies and consistency checking based on the capabilities of attached reasoners. Following the Web browser-approach, it reflects the characteristics of OWL being a language for the Semantic Web.

All ontology editing in SWOOP is done inline. Based on its HTML renderer, SWOOP uses different colour codes and font styles to emphasize ontology changes. Undo/redo options are provided with an ontology change log and a rollback option.

Some of the core features of SWOOP are the debugging features for OWL ontologies, exploiting features of OWL reasoners (in this case Pellet). This includes for example the automatic generation of explanations for a set of unsatisfiable axioms (e.g. for a particular class).

SWOOP can be characterized as a “pure” OWL tool, focusing on core features of the language rather then on general ontology development tasks. The tool has to offer additional features such as a basic version control, it does not include a couple of typical functionalities going beyond OWL editing, such as the integration or import of external (non-OWL/RDF-) sources.

4.1.2.6 OntoStudio

OntoStudio® is a commercial product of ontoprise[37]. It is a the front-end counterpart to OntoBroker®, a fast datalog based F-Logic inference machine. Consequently a focus of the OntoStudio® development has been on the support of various tasks around the application of rules. This includes the direct creation of rules (via a graphical rule editor) but also the application of rules for the dynamic integration of datasources (using a database schema import and a mapping tool).

OntoStudio® is available with a main memory- or database-based model, is therefore scaleable and is thus suitable for modelling even large ontologies. Based on Eclipse OntoStudio® provides an open framework for plugin developers. It already provides a number of plugins such as a query plugin, a visualizer and a reporting plugin supporting the Business Intelligence Reporting Tool (BIRT).

Just like TopBraid Composer™, OntoStudio® is implemented as IDE-application using the Eclipse platform with all the advantages such as the plugin concept.

OntoStudio® is tightly coupled to F-Logic (resp. its proprietary XML serialization OXML); the import and export from/to OWL/RDF is restricted mainly to concepts which can be expressed in F-Logic. Despite some minor syntactical details the Ontoprise F-Logic dialect conforms semantically to the F-Logic definition (Kifer, M. et al., 1995). Ontoprise is in close contact with the F-Logic forum to work on future versions of F-Logic and further standardization efforts.

OntoStudio® offers a graphical and a textual rule editor as well as debugging features as well as a form-based query-editor. It also includes a graphical editor for the creation and management of ontology mappings including conditional mappings, filters and transformations. Thus OntoStudio® takes advantage of the capabilities of F-Logic regarding rules (such as the support for function symbols).

4.1.2.7 Other Tools

In the last decade several tools that allow dealing with ontologies have appeared, some of them as plugins of ECLIPSE. Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It can be used to develop applications in Java and, by means of various plug-ins, other programming or modeling languages, such as UML and OWL. Some plug-ins that work over Eclipse deal with ontologies are: DOME[38], which is a programmable XML editor, the NeOn toolkit[39], which is an ontology engineering environment that provides comprehensive support for the ontology engineering life-cycle, Onotoa[40], which is an ontology editor for topic maps and has a graphical UML-like interface, and MatchIT[41], which automates and facilitates schema matching and semantic mapping between different Web vocabularies.

---

[37] http://semanticweb.org/wiki/OntoStudio

[38] http://dome.sourceforge.net/

[39] http://neon-toolkit.org/wiki/Main_Page

[40] http://onotoa.topicmapslab.de/

[41] http://www.revelytix.com/matchit.php

An exhaustive and updated list of the available ontology tools can be found in TechWiki[42].

4.2 Ontology Evaluation

Although there is a lack of systematic ways to develop ontologies, significant efforts have been made to evaluate them. Methodologies, metrics and techniques have been created to evaluate the quality of ontologies based on their goals, content, or usability. Hartmann et al (Hartmann and others 2004) propose a classification grid for ontology evaluation that takes into account the goal of an ontology, supported functions, application, usability, usefulness, and types of users (application users or knowledge engineers). Other ontology evaluation techniques are based on the kind of information the techniques evaluate (i.e., lexical, syntactic, hierarchy of the taxonomy, non-taxonomic relationship types, context or application level or structure). These techniques are based on: 1) comparing an ontology to a "golden standard"; 2) using an ontology in an application and evaluating its results; 3) comparing an ontology to a source of data about the domain to be covered by the ontology; and 4) assessing the ontology by humans on how well the ontology meets a set of predefined criteria, standards, and requirements (Brank, Grobelnik, Mladenic 2005).

The following are the most well-known metrics of ontology evaluation:

1 Burton-Jones et al (Burton-Jones and others 2005) provide a general-purpose evaluation metrics based on semiotics. It assesses the quality of an ontology on syntactic, pragmatic, semantic, and social aspects. It does not deal with the population of the ontology (instances), nor allow the method to be expanded easily, nor provide any facility for ontology searching.

2 Instance metrics of OntoQA (Tartir and others 2005) evaluate how well an ontology captures the real world. Instance metrics are divided into Knowledgebase metrics, which are applied to the set of instances as a whole, and Class metrics, which have a value for each particular class of the ontology and describe the way in which each class is used within the knowledge base.

3 A Peer-Review approach (Supekar 2004) incorporates subjective comments of ontology users and enables users to provide qualitative ratings on the ontology content. A metadata ontology provides: 1) source metadata, provided by the ontology authors and generated by the ontology-development tools, and 2) third-party metadata, provided by ontology users which include peer reviews of ontologies, usage and experience information and rating.

4 The theoretical framework of O2 (Gangemi and others 2005) specifies the used metrics. O2 contains a set of meta-ontologies that facilitate describing the ontologies, metrics, how to calculate the metrics, and the relationship between them. The formalization of the metrics could improve its extensibility and facilitate the implementation of the entire approach.

5 Alani and Brewster (Alani and Brewster 2006) support ontology search by AKTiveRank, which is a prototype for ranking ontologies based on the analysis of their structures. In the ontology search activity, users provide a set of keywords to an ontology search engine, which should rank the ontology according to the users' needs. AKTiveRank contains metrics to perform such a ranking activity. It uses Class Match, Density, Semantic Similarity and Betweenness measures to evaluate how close an ontology is to a set of keywords provided by the user.

6 Sabou et al (Sabou, Lopez, Motta 2006) analyze whether actual evaluation ontology techniques may be used for ontology selection per se, or if the requirements of ontology

[42] http://techwiki.openstructs.org/index.php/Ontology_Tools

selection make the evaluation techniques insufficient. They study, from a practical point of view, how ontology evaluation may help ontology search and that ontology selection and evaluation are complimentary.

4.3 Use of ontologies

Nowadays, one of the most popular of the repositories of knowledge about the real world are found in the form of ontologies, which are being developed at two levels (Guarino 1998): 1) individual domain ontologies that capture concepts about a particular application domain, and 2) upper level ontologies that contain massive amounts of knowledge about the real world and are domain independent. The most well-known upper level ontology is Cyc (encyclopedia) project (Lenat 1995), which is an ambitious attempt to capture common sense knowledge about the world and encode it in a knowledge base. ResearchCyc[43], a version of Cyc has been made available for use by the scientific community.

An ontology is a specification of a representational vocabulary for a shared domain of discourse (Gruber 1993b). It should be a way of describing one's world (Weber 2010) and generally consists of terms, their definitions, and axioms relating them. Ontologies provide a shared and common understanding of a domain that can be communicated between people and applications (Gruber 1993b).

Ontologies are intended to provide an "easy to re-use" library of class objects for modeling problems and domains. The ultimate goal is to construct a library of ontologies which can be reused and adapted to different general classes of problems and environments (Uschold and Gruninger June, 1996). Providers of such libraries have a huge impact on information exchange, and the assets they provide can be used to facilitate the integration and translation processes between people and systems (Fensel 2004).

The Semantic Web, the next generation of the World Wide Web, is intended to enable more intelligent use of data for effective electronic integration, interoperability and collaboration. The Semantic Web depends upon the ability to manage, integrate, and analyze data and is driven by the role of semantics for automated approaches to exploiting Web resources (Lee, Hendler, Lassila 2001). To address this need, ontologies are being developed to serve as surrogates for semantics.

There are a number of challenges associated with ontology development and re-use. Each ontology is specific to a domain and is difficult to create because it requires a thorough knowledge of the domain to be described (Herman 2007).

Ontologies are complex, with large-scale ontologies requiring a collaborative and on-going community effort from knowledgeable people. Although ontologies should be shared and reused, this is difficult when different domain experts develop them and the domains themselves change (e.g. business) (Kim and Sengupta 2007). However, applications can also be developed with very small ontologies (Herman 2007). Ontologies may be culture-specific instead of being reusable across cultures.

The most well-known domain ontologies were developed for the DAML ontology library (www.daml.org) with over 200 application domains, such as furniture, movies, and illnesses. The most well known upper level ontology is Cyc.

[43] ResearchCyc ontology. Available from http://research.cyc.com (accessed 31 January 2011).

4.3.1 General Purpose Ontologies

4.3.1.1 Knowledge Reuse

Knowledge can, and often is, reused for a particular task. The knowledge contained in upper level ontologies is used in information engineering, information management, interoperability, conceptual modelling, the Semantic Web and integration. Some examples of their successful use are:

- To improve communication between different agents (persons or programs), provide support in the communication language or facilitate consensus among different collectives. One application is for portable devices (PDAs), which allows elderly people's health to be monitored remotely and continuously (Bagüés and others 2003). The system uses an operational ontology that allows the agents to communicate between themselves at a semantic level.
- To support the integration of different data sources. To this end, a task ontology is used to solve semantic inconsistency problems when global queries are being translated into local queries (Ras and Dardzinska 2004).
- To establish interoperability between different applications. For example, a domain ontology is used to establish dynamic interoperability between software agents (Embley 2004).
- To support natural language interpretation. Kowledge of the Cyc ontology is applied to solve interpretation problems in natural language and automatic translation (Mahesh and others 1996). Next sub section explains in detail the Cyc contology.
- To give semantic content to web pages. Examples are given (Wollersheim and Rahayu 2002) in which a medical ontology is used to create a dynamic ontology that classifies the concepts of medical web pages, and an ontology is used to increase the performance of queries related to telephone directory websites (Guarino, Masolo, Vetere 1999).
- To validate conceptual schemas, ontologies can improve the validation process (Shanks, Tansley, Weber 2003) and (Guarino and Welty 2002); for example, Ontoclean validates other ontologies in its creation process.

Existing methodologies for querying ontologies are often inadequate because they fail to take context into account. Ontologies, however, are intended to be surrogates for context.

4.3.1.2 The Cyc Ontology

The Cyc[44] ontology is a knowledge repository developed to capture and represent common sense. It contains more than 2.2 million assertions (facts and rules) describing over 250,000 terms, including 15,000 predicates. A full version of Cyc, ResearchCyc[45], contains both intensional information (entity types, relationship types, integrity constraint) and extensional information (representation of individuals and their relationship to space, time and human perception). Depending on the abstraction level of the knowledge, it can be classified in several layers as (see Fig 4.1. ResearchCyc Layer Structure):

- Upper Ontology: This represents very general concepts and relationships between them. For example, it contains assertions such as every event is a temporal thing, every temporal thing is an individual, and every individual is a thing. Thing is ResearchCyc's most general concept.

[44] Cyc: http://www.cyc.com (acceded 31 January 2011)

[45] ResearchCyc ontology. Available from http://research.cyc.com (accessed 31 January 2011).

- Core Theories: These represent general facts about space, time, and causality and are essential to almost all common sense reasoning. Examples include geospatial relationships, human interactions and everyday items and events.
- Domain-Specific Theories: These are more specific than core theories and deal with special areas of interest such as military movement, the propagation of diseases, finance, and chemistry.
- Facts: These represent extensional information, also known in ResearchCyc as ground-level facts.

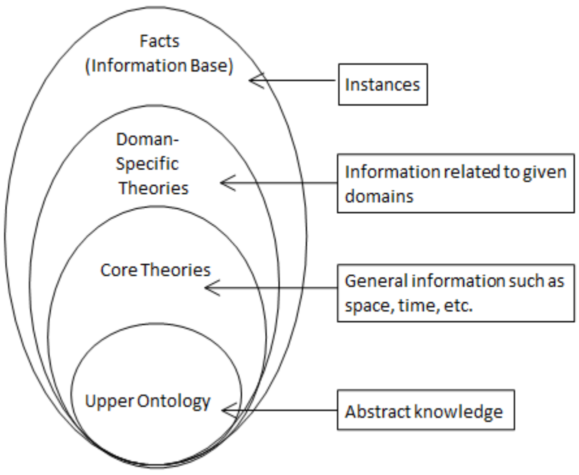


Fig 4.1. ResearchCyc Layer Structure

The first three layers describe intentional information (conceptual information) and the last one extensional information (facts). The general knowledge of ResearchCyc covers a broad range and can be classified as:

- Temporal knowledge describes the temporality of the concepts and their temporal relationships, such as something happens before something else.
- Spatial knowledge describes spatial properties of concepts such as the superposition of objects, connection, nearness and location, and part of relationships.
- Event information describes the most common events that can happen, the actors involved in the events, and their constraints.
- Geography information describes the geographic area of the concepts.
- General information includes, for example, emotion information.

A sample fragment from ResearchCyc is shown in Fig. 4.2. Cars are represented by the concept Automobile in Cyc. This concept participates in 315 constructions, including: subtypes, instances, relationship types, heuristics and constraints. There are over 300 relationships involving Automobile. A selection is shown in the figure.

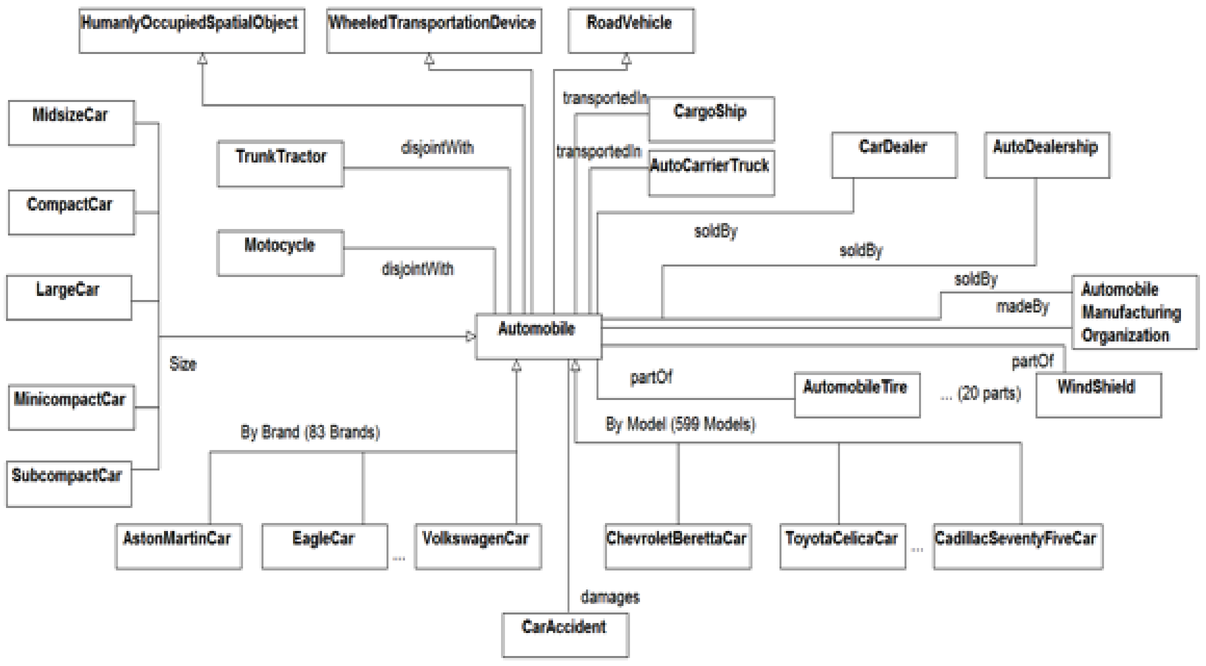


Fig4.2. Car Information from ResearchCyc

It is hard to use a knowledge base as large as ResearchCyc. The main problem is to discover whether the information to be looked for is defined in the ontology. Doing so manually is tricky because ResearchCyc has only a textual interface accessible by using a browser. It does not provide any facility to query and understand its knowledge. The deficiencies in the linguistic knowledge of ResearchCyc make the searching process even more difficult. Even if one is able to find the knowledge searched for, the problem is the large amount of knowledge retrieved. This makes it impossible to automate any process without using heuristics to automatically discard the information that is irrelevant for a particular context or to infer its semantics.

For example, if we are searching for biological knowledge, we know it should be an instance of generalMicrotheory or VocabularyMicrotheory. Hence, we can search their instances to find a microtheory (Mt) that deals with biological knowledge (BiologyMt). After locating the microtheory that contains the relevant knowledge, we would like to know to what extent the domain is represented in ResearchCyc. To determine this, one must take into account the microtheory (BiologyMt) and its super microtheories (BiologyVocabularyMt and BiochemistryMt). The definition of a local taxonomy of the selected microtheory is useful.

For inferences in ResearchCyc, the queries are executed using a microtheory as a context. It is, therefore, important to identify the correct microtheory for each query. Executing a query using a wrong Mt means that a query that may be answered using the ResearchCyc knowledge will have no results. For example, the query "in which city the liberty bell is located" (represented as (#$objectFoundInLocation #$LibertyBell ?CITY)) has no answers under the BaseKB microtheory. However, if carrying out the same query using the CurrentWorldDataCollectorMt Mt, returns Philadelphia as a result. Unfortunately, there is no Mt that fits all queries because the correct microtheory depends on the context of a query. For example, a query that deals with today's facts may need the CurrentWorldDataCollectorMt, and a query that deals with linguistic information may require EnglishMt microtheory. This problem worsens in web queries because the context of a web query cannot be obtained automatically from its terms. Therefore, even when ResearchCyc has relevant knowledge for a query, it may not be possible to retrieve it, because one cannot figure out on which microtheory to focus.

The amount of knowledge included in ResearchCyc should be simplified when we want to use it automatically. The ResearchCyc ontology contains 82 relationship types which means that an object is part of another object (partOf), such as subOrganization or capitalCity relationship types. One heuristic is to not know the exact semantics of each, but to treat them as simple partOf relationships.

4.3.2 Domain ontologies in the context of CSCL

Nowadays we can find thousands of different domain ontologies. Finding the right ontology has become a challenge and several search engines, such as swoogle[46], have appeared in order to facilitate the ontology search.

In the particular field of eLearning, quite a few ontologies (Wilson 2004) (Babic, Wagner, Paralic 2008) and related standards (Berlanga and Garcia 2005) concerning the representation of CSCL have been defined so far. Representative approaches include (Babic, Wagner, Paralic 2008) that use a combination of a general domain ontology describing the common semantics needed for the implementation of a collaboration environment with several domain ontologies that are used to provide a framework for end-user tools. Barros et al (Barros and others 2002; Barros, Mizoguchi, Verdejo) propose to use the actions performed in the collaborative learning system so as to build a high-level representation of the process of collection and analysis of the interaction data. In (Inaba and others 2000) a theory-oriented interaction analysis approach based on theories of collaborative learning is provided. However, the social processes happening behind real collaborative learning practices are very complex and subjective and thus they fall far from a holistic view proposed by standards and ontologies (Sicilia and others 2009). As (Babic, Wagner, Paralic 2008) states, with a well-defined ontology structure, CSCL can accumulate the knowledge representation of learning objects, including participant background, group information, instruction designs, learning activities, learning outcomes, etc.

In order to specify the collaboration activities that occur during the learning experience, in addition to the actual CSCL ontologies, we can use some of the actual specifications. Related to this project, some of the most relevant specifications / standards are:

4.3.2.1 LOM

The *IEEE 1484.12.1-2002 Standard for Information Technology- Education and Training Systems-Learning Objects and Metadata*, commonly known as IEEE LOM[47] is a standard that allows describing learning resources by a set of metadata. The main objective or IEEE-LOM is to improve discovery, management, sharing and reusability of learning resources within and between different repositories. The standard has its origins in the metadata specifications proposed by IMS and ARIADNE (Alliance for Remote Instructional Authoring and Distribution Networks for Europe) and is compatible with Dublin Core[48], which is another of the relevant metadata specifications.

The metadata defined in the standard are classified within nine categories according their nature. These categories are:

1. General: general information about a learning resource such as author, name and keywords,

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

[47] http://ltsc.ieee.org/wg12/

[48] http://dublincore.org/

2. Lyfe cycle: information about the history and actual state of a learning resource,
3. Meta-metadata: metadata about the metadata of the learning object,
4. Technical: technical information about the resource, such as the kind of format (txt, html, powerpoint) the resource uses,
5. Educational: characteristics related to the pedagogical and educational aspects of the resources,
6. Rights: information about the rights of the resources and conditions to use them,
7. Relation: allows to define new relations between different learning objects,
8. Annotation: includes information about the comments about the learning objects written by the academic staff,
9. Classification: describes a resource according a given classification criteria.

4.3.2.2 SIOC

The SIOC initiative (Semantically-Interlinked Online Communities)[49] aims to enable the integration of online community information. Online community sites (weblogs, message boards, wikis, etc.) contains a valuable source of information and are candidates to search when we need some information. However, online community sites are like islands without bridges connecting them. SIOC is an attempt to link these online community sites, by using Semantic Web technologies to describe the information that communities have about their structure and contents, and to find related information and new connections between content items and other community objects.

SIOC provides a Semantic Web ontology for representing rich data from the Social Web in RDF: the SIOC Core Ontology. It is the foundation for Semantically-Interlinked Online Communities and can be used to express information contained within community sites in a simple and extensible way. The SIOC ontology was recently published as a W3C Member Submission[50] .

SIOC is commonly used in conjunction with the FOAF vocabulary for expressing personal profile and social networking information. SIOC enables semantic applications to be built on top of the existing Social Web.

4.3.2.3 FOAF

FOAF [51] stands for *Friend Of A Friend* and is an specification that describes a language devoted to represent the linking information of people and information using the Web. Regardless of whether information is or the format the data is structured with. It is a recommendation of W3C in constant evolution since its creation (mid-2000). It has a stable core of classes and properties that will not be changed, while new terms may be added at any time.

FOAF describes the world using simple ideas inspired by the Web. FOAF descriptions are published as linked documents in the web, by using RDF/XML or RDFa syntax. In its descriptions, there are only various kinds of things, which are called classes, and links, which are called properties. FOAF allows describing people, groups and documents. Main FOAF terms are grouped in the following categories:

---

[49] http://sioc-project.org/

[50] http://www.w3.org/Submission/2007/SUBM-sioc-spec-20070612/

[51] http://www.foaf-project.org/

- Core: describe characteristics of people and social groups that are independent of time and technology.
- Social Web: describe internet accounts, address books and information related to social web activities.
- Linked Data Utilities: FOAF is part of the Linked Data community[52] and, therefore, needs to establish a simple factual data via a networked of linked RDF.

4.3.2.4 MOAT

MOAT[53], which stands for Meaning Of A Tag, is a framework that allow giving semantic to tags.

Tags are widely used, but lacked of a machine-understandable meaning. The problem comes from the facts that people can use tags that have different meanings depending on the context, but can also use different tags to express the same thing. Since tags are not explicitly related to each other, it is very difficult to find out when two different tag have the same meaning.

MOAT aims to solve this problem by providing a way for users to define meaning(s) of their tag(s) using URIs of Semantic Web resources (such as URIs from DBpedia, geonames ... or any knowledge base).

4.3.2.5 SKOS

SKOS[54] is an area of work developing specifications and standards to support the use of knowledge organization systems (KOS) such as thesauri, classification schemes, subject heading systems and taxonomies within the framework of the Semantic Web. SKOS provides a standard way to represent knowledge organization systems using RDF. Its Specifications are currently published as W3C Recommendations, which means that they are in stable state.

Even though SKOS is neither closely related to eLearning nor to CSCL, it is worth to take it into account in this project. The reason is that, since it allows representing knowledge and its organization, it is used for most of the specifications related to semantic web and Linked Data. In our context, for example, SKOS is used by SIOC in order to define the topic of the posts.

4.4 Chapter summary

In this chapter we have shown how to develop, evaluate and use ontologies in a general sense. We have motivated the need for better methods and tools for a systematic way to develop ontologies since most ontologies are still developed manually. Current support for systematic development ontologies is found mainly in ontology libraries, pre-existent large ontologies and ontology reasoners. Also, well-known ontology development tools exist in the market, such as Protégé and Altova, as well as tools that deal with ontologies, such as Jena. These tools and others are presented and described. Evaluation is seen as critical during the development to assure quality ontologies. Most evaluation techniques are basedon the use of metrics. At the second half of the Chapter we present the use of ontologies mainly at twolevels: Upper level ontologies with massive amounts of knowledge about the real world and individual domain ontologies that capture concepts about a particular application domain. A representative example of the former is the Cyc ontology. The latter is presented in the

[52] http://linkeddata.org/

[53] http://moat-project.org/

[54] http://www.w3.org/2004/02/skos/

specific domain of Web collaboration which is the context of ALICE forums. Representative efforts in this domain are SIOC and FOAF, which provide a Semantic Web ontology for representing rich data from the Social Web and a language devoted to represent the link of people and information using the Web.

5 Methodologies and techniques for knowledge modelling and representation of collaborative sessions

Up to now there has been a great effort in the semantic web community in order to provide specifications, standards and ontologies to facilitate semantic processes in the web. Our aim is to take advantage of these works as much as possible for the purpose of modelling and representing information and knowledge of collaborative learning in the context of online forums. To this end, an ontological framework has been created. This framework allows representing information about collaboration activities realized using online forums by means of ontologies, aligns such ontologies with others in order to facilitate the importation of data from and to other semantic sources and facilitates the population of its ontologies from different kinds of web forums.

In this project, the framework has been used by the Virtualized Collaborative Session system (VCS). Such system enables the virtualization of collaborative sessions, creating animated storyboards that reproduce collaboration sessions that have been taken within web forums of virtual classrooms. After integrating the ontological framework with VCS we proved that the information from the framework was enough to create animated storyboards. However, we also realized that some information about the opinion (or the sentiment) of each user would be very useful in improving the final result. The problem was that the face of the students in the animated storyboard was always the same, motionless even when their text denoted angriness or unhappy mood. Therefore, we decided to present a possible extension of the framework in order to include information about the sentiment and opinion of users when collaborating. Such addition has been done as a proof of concept and after the experimentation, so no details about its implementation or validation is addressed in this document.

This Chapter is structured as follows: first section will create an ontological framework that represents the collaboration activities that has been realized using online forums. The ontology of the framework has been aligned with SIOC, FOAF and SKOS in order to take into account other sources of data. A possible extension of the framework to represent sentiment and opinion information is also presented. Second section shows how the information generated in collaborative learning forums can be captured and classified at several description levels. This fact can significantly improve the way a collaborative system used for learning and instruction can collect all the necessary information produced from the user-user and user-system interaction in an efficient manner. The ultimate aim is to provide an efficient and robust computational approach that enables the effective collection and classification of data from ALICE forums. Finally, third section presents the experiment that has been conducted in order to validate the completeness and usefulness of the created framework.

5.1 An ontological framework for representing collaborative learning sessions within forums

This section presents an ontological framework created with the purpose of representing information from collaborative sessions, how its ontology has been integrated with relevant specifications such as SIOC and how the ontology can be automatically populated from web forums. The created ontology is named Collaborative Session Conceptual Schema (CS^2) and allows for representing the collaborative sessions where several actors have enjoyed in their learning experiences within forums. This representation in common format serves to input the virtualization of the collaborative sessions (VCS) (see Figure 5.2 for examples of web-based forums).

SIOC and other specifications define some concepts that are relevant in the ontology domain. In order to improve the generalization of our approach and provide operability with the most prominent standards and specifications we should align the RDF version of our ontology to the RDF versions of such standards/specifications. With that objective in mind, CS^2 can be stored or imported from files in CSML format (Collaborative Session Markup Language), which is in turn based on the RDF representation for the CS^2 ontology but aligned with SIOC ontology. We can see in figure 5.1 the main class structure of the SIOC core ontology.

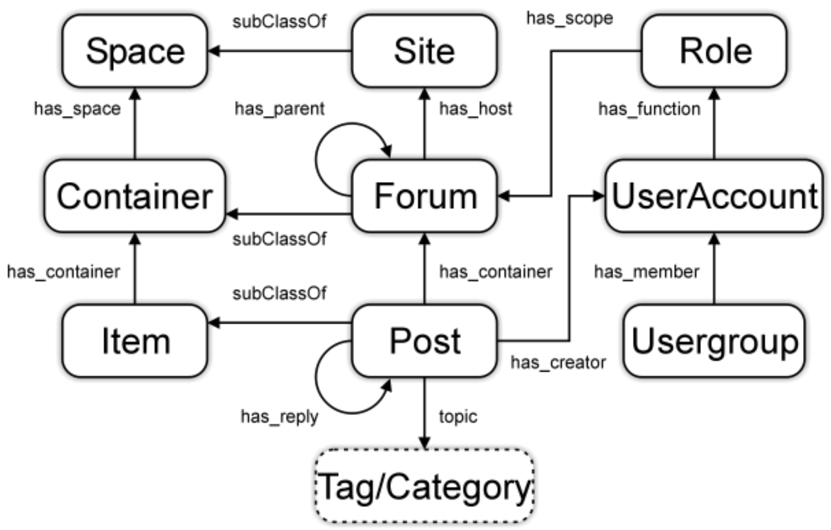


Fig. 5.1. SIOC Core Ontology Classes

The objective is to create an ontology that can be populated from either forums or the specification of forums written in some of the aligned formats, such as SIOC. In order to do so, we implemented some facilities that help the process of converting the information of particular kind of forums to CSML format. Figure 5.2 shows the CS^2 ontology in the context of the framework used to facilitate its population from forums and other specifications. Note that IWT forums and DF forums are the forums used for Intelligent Web Teacher (IWT) and the Discussion Forum (DF) (Caballé, Daradoumis, Xhafa Juan (2011)) at the Open University of Catalonia (UOC) respectively.

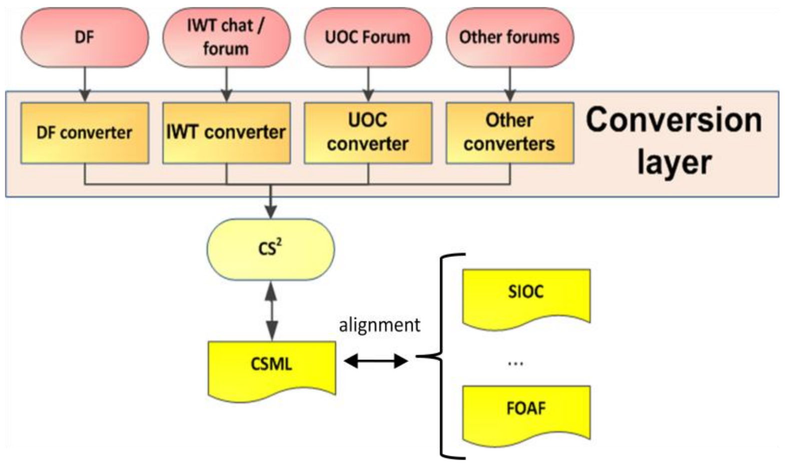


Fig 5.2. Framework for representing information related to Collaborative Learning Sessions from forums.

5.1.1 CS^2 Ontology: the Conceptual Model

As said before, CS^2 is based on SIOC specification so it contains some of the elements defined on this and other related specifications like FOAF (Friend of a friend) or Dublin Core. However, some classes and properties of SIOC specification are not potentially useful to our domain. In this subsection we enumerate the elements from SIOC that are applicable to our ontology. Obviously, the list of elements may (and surely will) change or grow in number as needed during the VCS system development.

CS^2 represents information about collaborative sessions. A collaborative session can be seen as a set of activities performed by several users playing several roles to achieve a common result. We are especially interested in the collaborative sessions that occur in a virtual environment, such as chats or forums. As we can see in figure 5.3, the main entity of the CS^2 ontology is *CollaborativeSession*, which occurs within a site. A list of users, represented by the class *UserAccount*, can collaborate in the session with different *roles*. Each piece of communication within a session is represented by a *post*, which is created by one of the collaborators and may be categorized. The posts are related between them in a threaded structure through the *replies* relation.

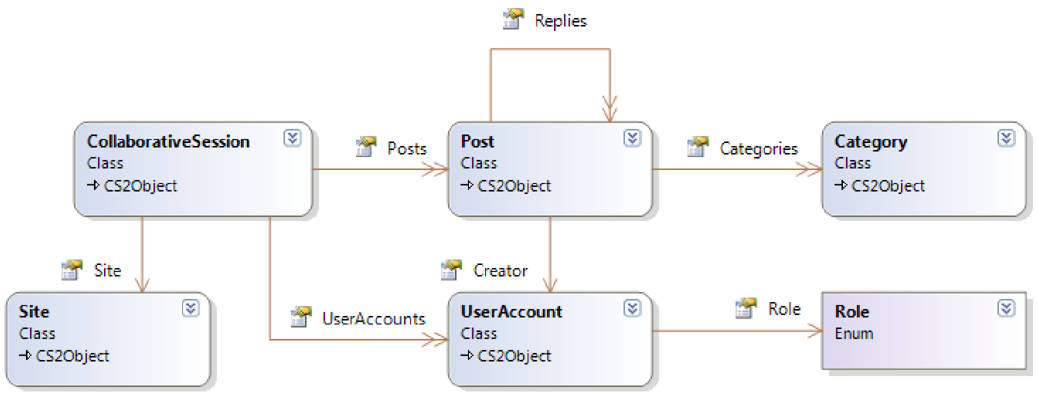


Fig. 5.3. Excerpt of the CS^2 Ontology.

In the following we present the structure of the ontology that allow storing and working with collaborative session data. In particular, figure 5.4 shows the class hierarchy of the ontology.

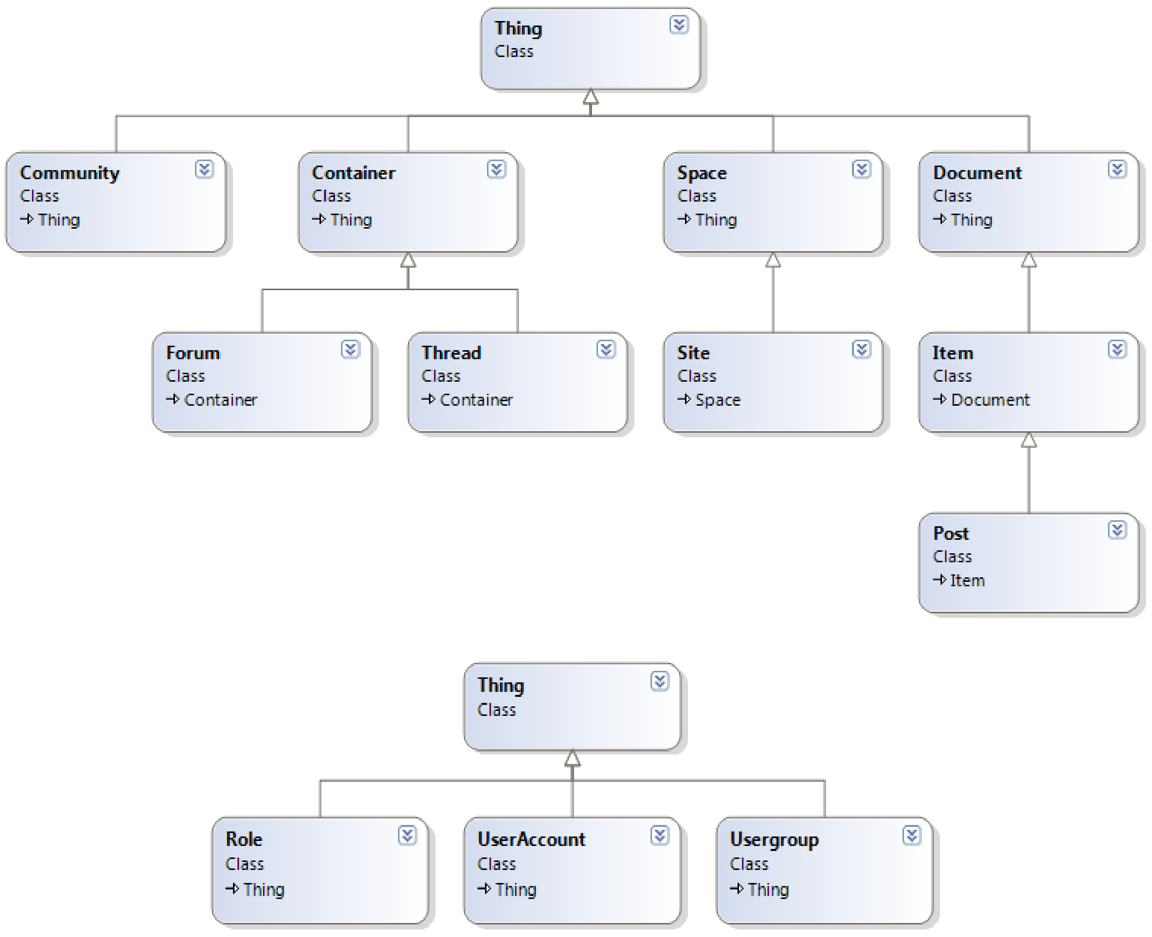


Fig. 5.4. CS^2 Class hierarchy

The next figures represent the relationships between the different classes of the ontology. In particular, figure 5.5 shows the main relationships related to *Spaces*. The main relations of *Sites* and *Users* are shown in figures 5.6 and 5.7 respectively.

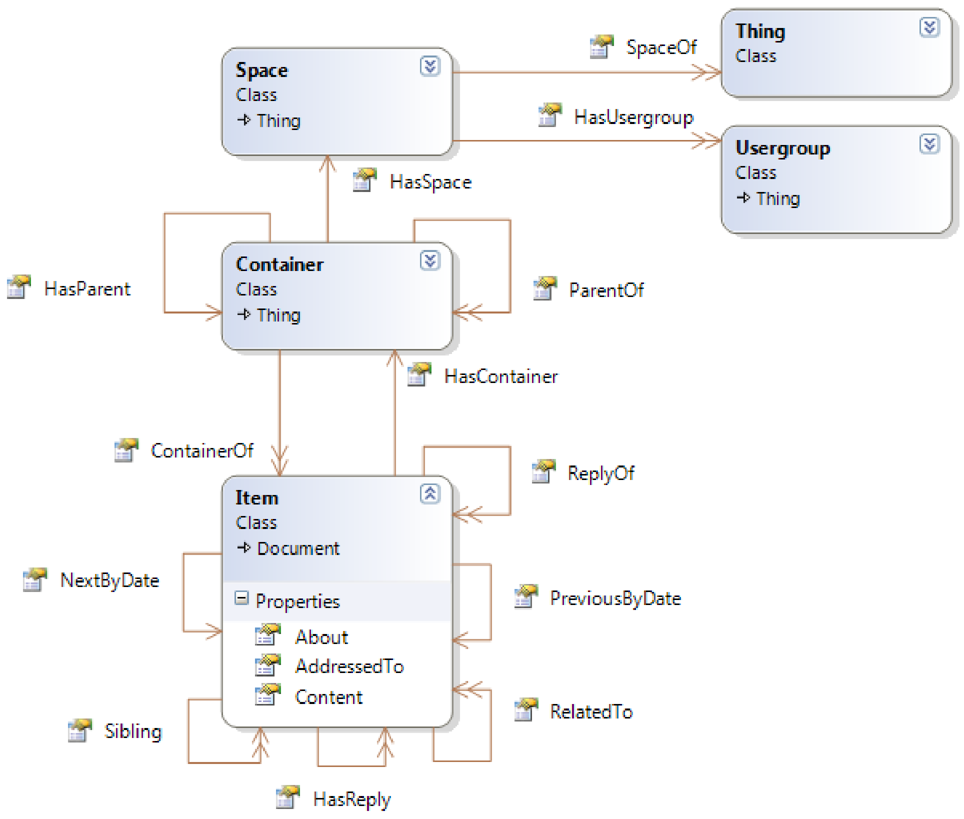


Fig. 5.5. Spaces Class and its relationship types

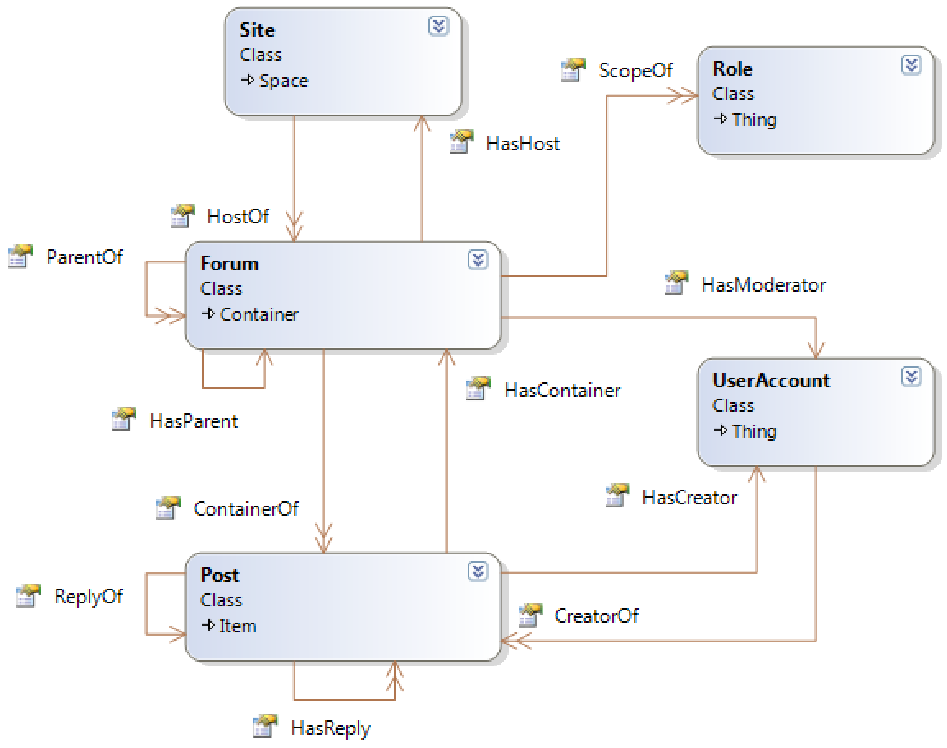


Fig. 5.6. Sites and its relationship types

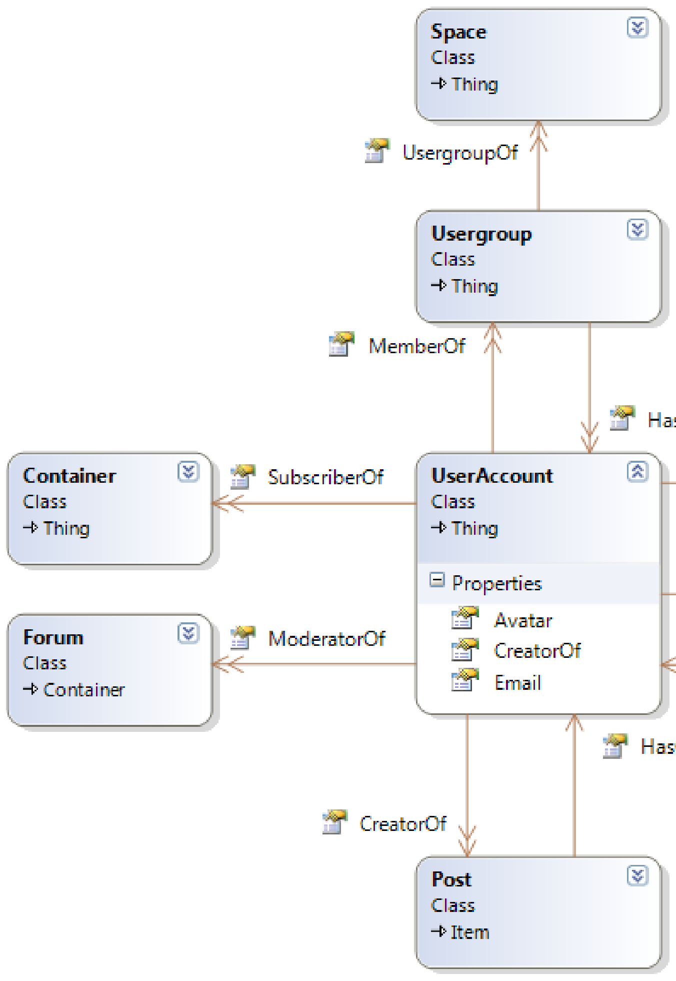


Fig. 5.7. Users and its relationship types in CS^2

5.1.2 Aligning CS^2 to other Relevant Specifications

As aforesaid, CS^2 should be aligned with SIOC specification so it contains some of the elements defined on this and other related specifications, such as FOAF or Dublin Core. As a RDF version of the CS^2 ontology, we decided to define a specific and purpose-delimited language that allows representing collaborative sessions and align them with the classes and properties from SIOC that are useful to the representation of collaborative sessions, called Collaborative Session Markup Language (CSML). In order to maintain compatibility with SIOC, the CS^2 conceptual model can be imported (and exported) from (and into) CSML.

Table 5.1: Relevant SIOC classes to represent collaborative sessions. These are the SIOC classes aligned to CSML.

Class	**Description**
Community	Community is a high-level concept that defines an online community and what it consists of.
Container	An area in which content Items are contained.
Forum	A discussion area on which Posts or entries are made.

Item	An Item is something which can be in a Container.
Post	An article or message that can be posted to a Forum.
Role	A Role is a function of a UserAccount within a scope of a particular Forum, Site, etc.
Space	A Space is a place where data resides, e.g. on a website, desktop, fileshare, etc.
Site	A Site can be the location of an online community or set of communities, with UserAccounts and Usergroups creating Items in a set of Containers. It can be thought of as a web-accessible data Space.
Thread	A container for a series of threaded discussion Posts or Items.
User Account	A user account in an online community site.
Usergroup	A set of UserAccounts whose owners have a common purpose or interest. Can be used for access control purposes.

As can be seen from table 5.1, SIOC Core ontology includes classes relevant to our purpose of modeling data coming from collaborative learning sessions (e.g. discussion forums), such as *Forum*, *Item*, *Post*, *Thread*, and so on. In addition, SIOC types ontology defines classes that represent different kinds of Containers, Forums, Items and Posts. Some examples of these classes are: *Address*

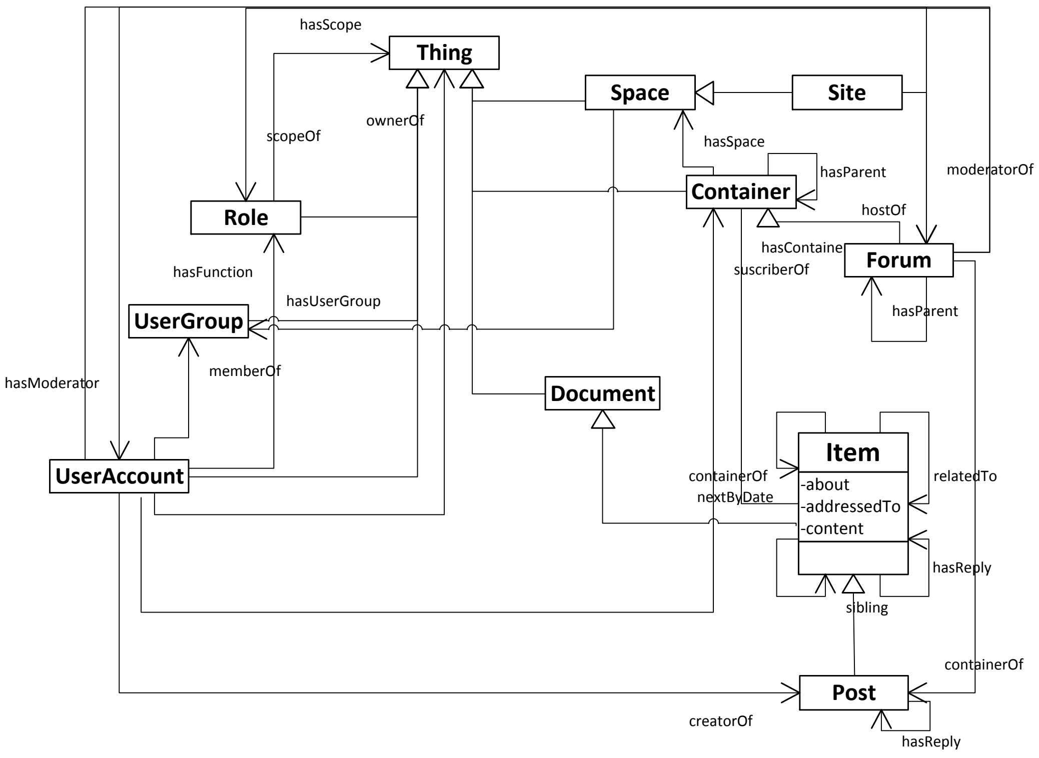


Fig. 5.8. Excerpt of the SIOC ontology aligned with CSML.

Book, *Image Gallery*, *Wiki*, *Chat Channel* and *Message Board*. For the sake of simplicity, we will work at the level of forums and posts at this stage, but will consider using their subclasses in further versions. The SIOC types ontology also includes two classes used to define post topics: *Category* and

Tag. *Category* is defined as a subclass of a SKOS Concept (see Section 4.3.2.5 for further information on SKOS). .

For simplicity, literal topics will be used in CSML, but having the possibility to extend the language in the future with these two classes. On the other hand, *User* class from SIOC specification is not used as it is deprecated: *UserAccount* is used instead

SIOC specification contemplates using some elements from other ontologies, such as FOAF or SKOS. Moreover, some of the SIOC elements are defined as subclasses or sub properties of ontologies like FOAF or Dublin Core. For the sake of simplicity, CSML properties will adjust to the subset of SIOC properties defined in table 5.2. Therefore, we need to include in CSML concepts from other ontologies that will take into account these new mechanisms, such as the class *Person* from FOAF that can help describe elements of *UserAccount* type with properties such as *firstName* and *lastName*, and also some elements from Dublin Core Terms that must be considered to include (e.g. *date*, *title*, *description* or *subject*). We can see in figure 5.8 the main classes and properties of SIOC Core Ontology aligned with CSML. Note that, in order to improve readability, we have omitted inverse relationships in figure 5.8. Therefore, it should be assumed that there is an inverse relationship for each of the relationships drawn on the figure.

Table 5.2: SIOC properties aligned to CSML.

Property	**Description**
about	Specifies that this Item is about a particular resource, e.g. a Post describing a book, hotel, etc. domain: http://rdfs.org/sioc/ns#Item
avatar	An image or depiction used to represent this UserAccount. domain: http://rdfs.org/sioc/ns#UserAccount subPropertyOf: http://xmlns.com/foaf/0.1/depiction
container of	An Item that this Container contains. inverseOf: http://rdfs.org/sioc/ns#has_container domain: http://rdfs.org/sioc/ns#Container range: http://rdfs.org/sioc/ns#Item
content	The content of the Item in plain text format. domain: http://rdfs.org/sioc/ns#Item range: http://www.w3.org/2000/01/rdf-schema#Literal
creator of	A resource that the UserAccount is a creator of. inverseOf: http://rdfs.org/sioc/ns#has_creator domain: http://rdfs.org/sioc/ns#UserAccount
email	An electronic mail address of the UserAccount. domain: http://rdfs.org/sioc/ns#UserAccount
function of	A UserAccount that has this Role. inverseOf: http://rdfs.org/sioc/ns#has_function domain: http://rdfs.org/sioc/ns#Role

Property	Description
has container	The Container to which this Item belongs. inverseOf: http://rdfs.org/sioc/ns#container_of domain: http://rdfs.org/sioc/ns#Item range: http://rdfs.org/sioc/ns#Container
has creator	This is the UserAccount that made this resource. inverseOf: http://rdfs.org/sioc/ns#creator_of range: http://rdfs.org/sioc/ns#UserAccount
has function	A Role that this UserAccount has. inverseOf: http://rdfs.org/sioc/ns#function_of range: http://rdfs.org/sioc/ns#Role
has host	The Site that hosts this Forum. inverseOf: http://rdfs.org/sioc/ns#host_of domain: http://rdfs.org/sioc/ns#Forum range: http://rdfs.org/sioc/ns#Site
has member	A UserAccount that is a member of this Usergroup. inverseOf: http://rdfs.org/sioc/ns#member_of domain: http://rdfs.org/sioc/ns#Usergroup range: http://rdfs.org/sioc/ns#UserAccount
has moderator	A UserAccount that is a moderator of this Forum. domain: http://rdfs.org/sioc/ns#Forum range: http://rdfs.org/sioc/ns#UserAccount
has owner	A UserAccount that this resource is owned by. inverseOf: http://rdfs.org/sioc/ns#owner_of range: http://rdfs.org/sioc/ns#UserAccount
has parent	A Container or Forum that this Container or Forum is a child of. inverseOf: http://rdfs.org/sioc/ns#parent_of domain: http://rdfs.org/sioc/ns#Container range: http://rdfs.org/sioc/ns#Container
has reply	Points to an Item or Post that is a reply or response to this Item or Post. inverseOf: http://rdfs.org/sioc/ns#reply_of subPropertyOf: http://rdfs.org/sioc/ns#related_to domain: http://rdfs.org/sioc/ns#Item range: http://rdfs.org/sioc/ns#Item
has scope	A resource that this Role applies to. inverseOf: http://rdfs.org/sioc/ns#scope_of domain: http://rdfs.org/sioc/ns#Role
has space	A data Space which this resource is a part of. inverseOf: http://rdfs.org/sioc/ns#space_of range: http://rdfs.org/sioc/ns#Space

Property	Description
has usergroup	Points to a Usergroup that has certain access to this Space. inverseOf: http://rdfs.org/sioc/ns#usergroup_of domain: http://rdfs.org/sioc/ns#Space range: http://rdfs.org/sioc/ns#Usergroup
host of	A Forum that is hosted on this Site. inverseOf: http://rdfs.org/sioc/ns#has_host domain: http://rdfs.org/sioc/ns#Site range: http://rdfs.org/sioc/ns#Forum
id	An identifier of a SIOC concept instance. For example, a user ID. Must be unique for instances of each type of SIOC concept within the same site. range: http://www.w3.org/2000/01/rdf-schema#Literal
member of	A Usergroup that this UserAccount is a member of. inverseOf: http://rdfs.org/sioc/ns#has_member domain: http://rdfs.org/sioc/ns#UserAccount range: http://rdfs.org/sioc/ns#Usergroup
moderator of	A Forum that a UserAccount is a moderator of. inverseOf: http://rdfs.org/sioc/ns#has_moderator domain: http://rdfs.org/sioc/ns#UserAccount range: http://rdfs.org/sioc/ns#Forum
next by date	Next Item or Post in a given Container sorted by date. inverseOf: http://rdfs.org/sioc/ns#previous_by_date domain: http://rdfs.org/sioc/ns#Item range: http://rdfs.org/sioc/ns#Item
num views	The number of times this Item, Thread, UserAccount profile, etc. has been viewed. range: http://www.w3.org/2001/XMLSchema#nonNegativeInteger
owner of	A resource owned by a particular UserAccount, for example, a weblog or image gallery. inverseOf: http://rdfs.org/sioc/ns#has_owner domain: http://rdfs.org/sioc/ns#UserAccount
parent of	A child Container or Forum that this Container or Forum is a parent of. inverseOf: http://rdfs.org/sioc/ns#has_parent domain: http://rdfs.org/sioc/ns#Container range: http://rdfs.org/sioc/ns#Container
previous by date	Previous Item or Post in a given Container sorted by date. inverseOf: http://rdfs.org/sioc/ns#next_by_date domain: http://rdfs.org/sioc/ns#Item range: http://rdfs.org/sioc/ns#Item
related to	Related Posts for this Post, perhaps determined implicitly from topics or references. domain: http://rdfs.org/sioc/ns#Item range: http://rdfs.org/sioc/ns#Item

Property	Description
reply of	Links to an Item or Post which this Item or Post is a reply to. inverseOf: http://rdfs.org/sioc/ns#has_reply subPropertyOf: http://rdfs.org/sioc/ns#related_to domain: http://rdfs.org/sioc/ns#Item range: http://rdfs.org/sioc/ns#Item
scope of	A Role that has a scope of this resource. inverseOf: http://rdfs.org/sioc/ns#has_scope range: http://rdfs.org/sioc/ns#Role
sibling	An Item may have a sibling or a twin that exists in a different Container, but the siblings may differ in some small way (for example, language, category, etc.). The sibling of this Item should be self-describing (that is, it should contain all available information). domain: http://rdfs.org/sioc/ns#Item range: http://rdfs.org/sioc/ns#Item
space of	A resource which belongs to this data Space. inverseOf: http://rdfs.org/sioc/ns#has_space domain: http://rdfs.org/sioc/ns#Space
subscriber of	A Container that a UserAccount is subscribed to. inverseOf: http://rdfs.org/sioc/ns#has_subscriber domain: http://rdfs.org/sioc/ns#UserAccount range: http://rdfs.org/sioc/ns#Container seeAlso: http://rdfs.org/sioc/ns#feed
topic	A topic of interest, linking to the appropriate URI, e.g. in the Open Directory Project or of a SKOS category. subPropertyOf: http://purl.org/dc/terms/subject
usergroup of	A Space that the Usergroup has access to. inverseOf: http://rdfs.org/sioc/ns#has_usergroup domain: http://rdfs.org/sioc/ns#Usergroup range: http://rdfs.org/sioc/ns#Space

SIOC specification contemplates using some elements from other ontologies, moreover, some of the SIOC elements are defined as subclasses or subproperties of that ontologies like FOAF or Dublin Core.

In addition to the alignment with SIOC presented above, CS^2 includes concepts from other ontologies. For example FOAF defines some classes like *Person* that can help in better describe elements of *UserAccount* type with properties like name, *firstName*, *lastName*, etc. The property depiction also from FOAF, relates an instance with an image that depicts it (used in SIOC to define *UserAccount* avatar). Dublin Core Terms also defines some elements that must be considered to include in the future, like date, title, description or subject, the last one used to define topics associated to a post.

5.1.3 Populating the CS^2 ontology from Forums

Once the ontology has been created, the next problems to be faced are (i) how to process the information collected during the collaboration in order to facilitate its later analysis and make the

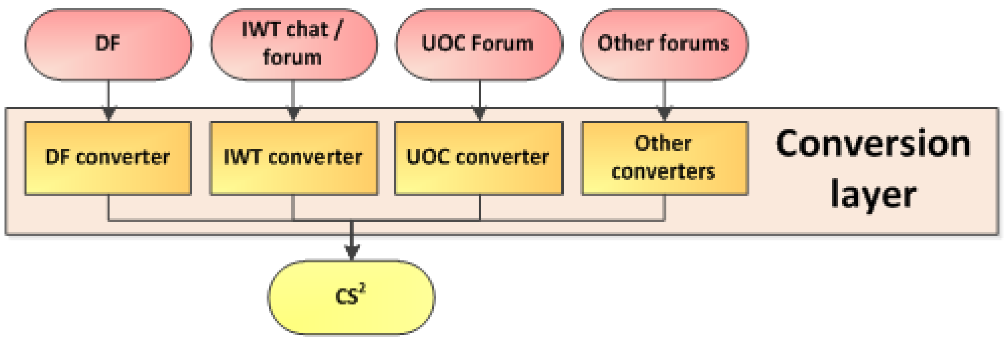


Fig. 5.9. Conversion Layer from different web forums such as Discussion Forum (DF), Intelligent Web Teacher (IWT) and forums form the Open University of Catalonia (UOC).

extracted knowledge available to the participants; (ii) how information should be analyzed and what kind of knowledge should be extracted to be fed back to the participants in order to provide the best possible support and monitoring of their learning and instructional processes. Section 5.2 proposes a solution to these problems by describing in detail a methodological approach of a process of embedding information and knowledge into collaborative learning applications in an efficient manner (Caballé et al 2010).

We now proceed with filling the ontology instances with the appropriate data collected and classified during the collaboration. As it is fully explained in Section 5.2, this data will be afterwards transformed into useful knowledge about what is happening during the collaboration by means of analysis techniques.

To this end, we base the data collection and classification into our ontology on the interaction occurred and registered in the context of online forums. The focus is on student interaction among peers driven by posts in online forums, which is the cornerstone of this approach. Participants need indeed to interact with each other to plan an activity, distribute tasks, explain, clarify, give information and opinions, elicit information, evaluate and contribute to the resolution of problematic issues, and so on.

The proposed Architecture defines a layer of converter components (see figure 5.9), each of which converts collaborative session data from different web forums into instances of CS^2 representing the same knowledge. Each converter will map the data from the corresponding data source into CS^2 entities, which at the end would be stored into a CSML. As we can see in figure 5.10, converters are defined as black boxes with a common interface that provides basically two interaction points:

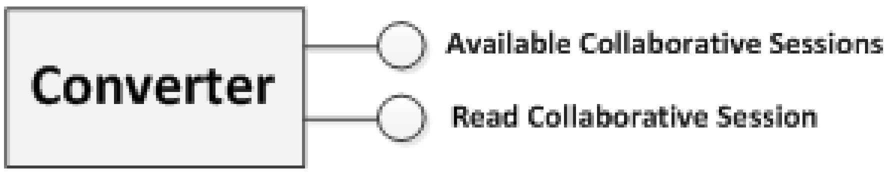


Fig. 5.10. The Converter Interface.

- *Available Collaborative Sessions*, which returns a list of available collaborative sessions on the data source to convert. It does not return all data from collaborative sessions, only descriptive information.
- *Read Collaborative Session*, which, given a collaborative session identifier, returns all the information of the corresponding collaborative session in the CS^2 ontology.

The conversion process done by each specific converter component can be viewed as a deterministic mapping between two data models (original data source schema and CS^2) following a set of predefined mapping rules. These rules will vary depending on the converter being developed.

Although all converters will have a common structure or share the same tasks (read data from data source, and create instances of CS^2 entities), the implementation of each one is dependent of the type of data source being used. Up to now, converters for DF and IWT forums have been implemented for reference and validation purposes.

5.1.4 Extending CS^2 with sentiment and opinion information

Using the basic information of CS^2 ontology we have been able to develop a system that converts the collaborative sessions of virtual classrooms to storyline learning objects (SLO), which are learning objects in a video format that reproduce the collaborations experienced in the classrooms. The next step is to add to CS^2 information that describes the opinions and sentiments associated to the users when collaborating. Doing so, we will be able to improve the existent services and add new functionalities to our system. For example, adding sentiment and opinion information to CS^2 will allow creating SLO that represent better the reality of the collaboration, by changing the voice tones or the

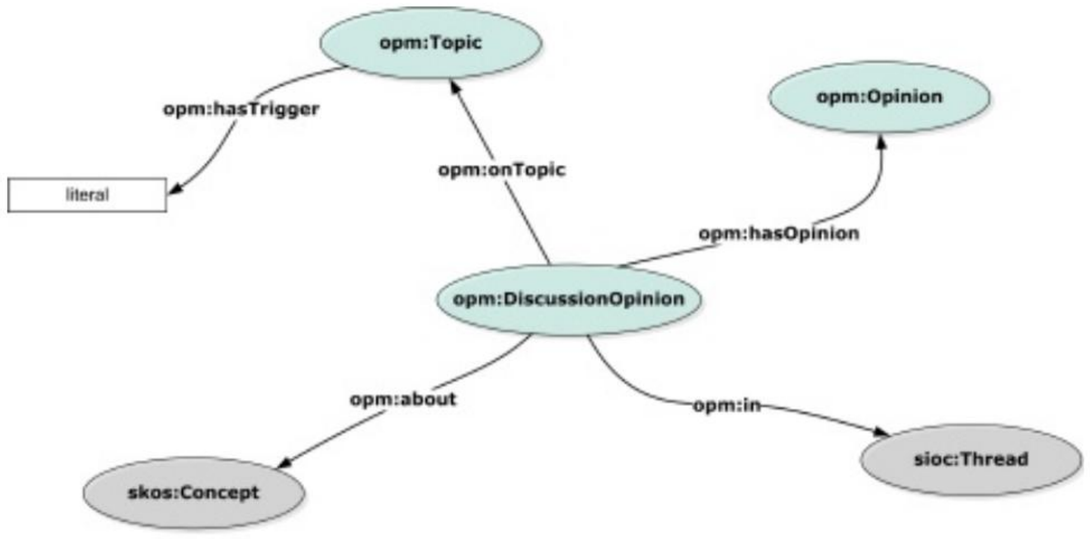


Fig. 5.11. Opinion Mining Core Ontology.

avatars according to the opinion or sentiment expressed for users when stating a given opinion. This information will also help the extraction of knowledge from the interaction data and a more personalized learning design, since the activities proposed to student may depend on the mood of the user and then present more easy activities to discouraged students in order to improve their self-esteem and therefore their motivation in the course.

Automatic opinion mining and sentiment analysis a computational discipline within the field of NLP, it consists of detecting fragments of text in which an opinion on a given matter is expressed or what sentiment was felt for the writer when writing the text. Automatically mining the opinions and sentiments expressed in a text is a complicated NLP task, one for which a range of strategies rooted in different approaches have been used. One type of strategy is based on information recovery techniques (Turney, 2002 and Read, 2004), which firstly identify a text's polarity and then its affective content. Such techniques are used to recover texts discerningly, based on their polarity. A second strategy type uses supervised learning and classification techniques, such as support vector machines (Corina and Vapnik, 2005) or latent semantic analysis (Deerwester et al., 1990), to develop statistical models for classifying texts according to emotions (Leshed and Kaye, 2006). The drawback to supervised learning is that relatively large quantities of tagged samples are required for model development.

In our case, we focus in the way of representing opinions and sentiments related to the contributions in collaboration sessions. Therefore, we will focus in the extension of CS^2 for representing such information. In that way we will be provide independency of the opinion/sentiment mining technique (or techniques) to be used.

Up to now different ontologies have been proposed to deal with opinion and sentiment information (Softic and Hausenblas, 2008) and KDO ontology[55]. In (Softic and Hausenblas, 2008) an approach to mine opinions from discussion forums was created. In that approach the forums were exported to SIOC format and thereafter linked with other datasets from DBpedia. In order to explicitly model the opinions in a discussion they created an ontology called "Opinion Mining Core Ontology" (see figure 5.11). This ontology uses SKOS to represent what a discussion is about and SIOC to indicate where the discussion has taken place. Even though it may seem a potential candidate to be reused in our context, we discarded this ontology since its representation is partial: the *opinion* class of the ontology is underspecified; it lacks concepts for representing sentiments and polarities of opinions.

Another potential candidate to reuse in our context is the KDO ontology (Thalhammer et al. 2011), which has been created in the context of the RENDER FP7 project. This project is still in course and focus on enabling and retrieval of knowledge diversity. The design of the KDO (see figure 5.12) builds on concepts and properties of SIOC and FOAF ontologies. The main concepts of this ontology are *Opinion*, *Sentiment*, *Polarity*, and *Bias*. An opinion is used to model the concept of opinion and can have one or more opinion expressions (*hasOpinionExpression*), can be linked to an emotion (*hasEmotion*), can mention named entities (*mentions*) and can have a sentiment (*hasSentiment*). *OpinionStatement* is used to indicate that an opinion (*hasOpinion*) has been stated in a post (*hasAgentOpinion*) by an agent (*isOpinionHeldBy*). Sentiments are linked to posts and opinions (*hasSentiment*) and may have polarities (*hasPolarity*), which can be positive, neutral and negative. A bias can be related to a SIOC post/space or an agent (*hasBias*), to one or more opinions (*relatedOpinion*) and to diferent biases (*relatedTo*).

[55] http://render-project.eu/resources/kdo/

As aforesaid, the KDO ontology is complete enough to represent opinions, sentiments and biases in the context of SIOC spaces and FOAF agents. Since our ontology is aligned with SIOC and FOAF we plan to use the KDO ontology to represent the opinions of collaborative sessions. In order to do so, we will link the KDO ontology with CS^2. Since the classes *Post* and *Space* of our ontology are equivalent to the same classes of SIOC, these classes will be used to indicate the origin of biases, opinions and sentiments. The class *UserAccount* will be used instead of *foaf:Agent* in order to specify the opinion holder. The use of the KDO ontology has been complemented with the Human Emotion Ontology (HEO) (see (Grassi, 2009)) that will allow to represent semantically the emotions related to each opinion.

5.2 A methodology to model and represent collaborative interaction data

So far we have described the knowledge our ontology needs to have and the ontologies we use in order to represent such knowledge. We now proceed with filling the ontology instances with the appropriate data collected and classified during the collaboration according to the classes and relationships of the ontology. The data collected will be afterwards transformed into useful knowledge about what is happening during the collaboration by means of analysis techniques (see sub Section 6.3.1 of D3.2.1).

To this end, we base the data collection and classification into our ontology on the interaction occurred and registered in the context of online forums. The focus is on student interaction among peers driven by posts in online forums, which is the cornerstone of this approach. Participants need indeed to interact with each other to plan an activity, distribute tasks, explain, clarify, give information and opinions, elicit information, evaluate and contribute to the resolution of problematic issues, and so on.

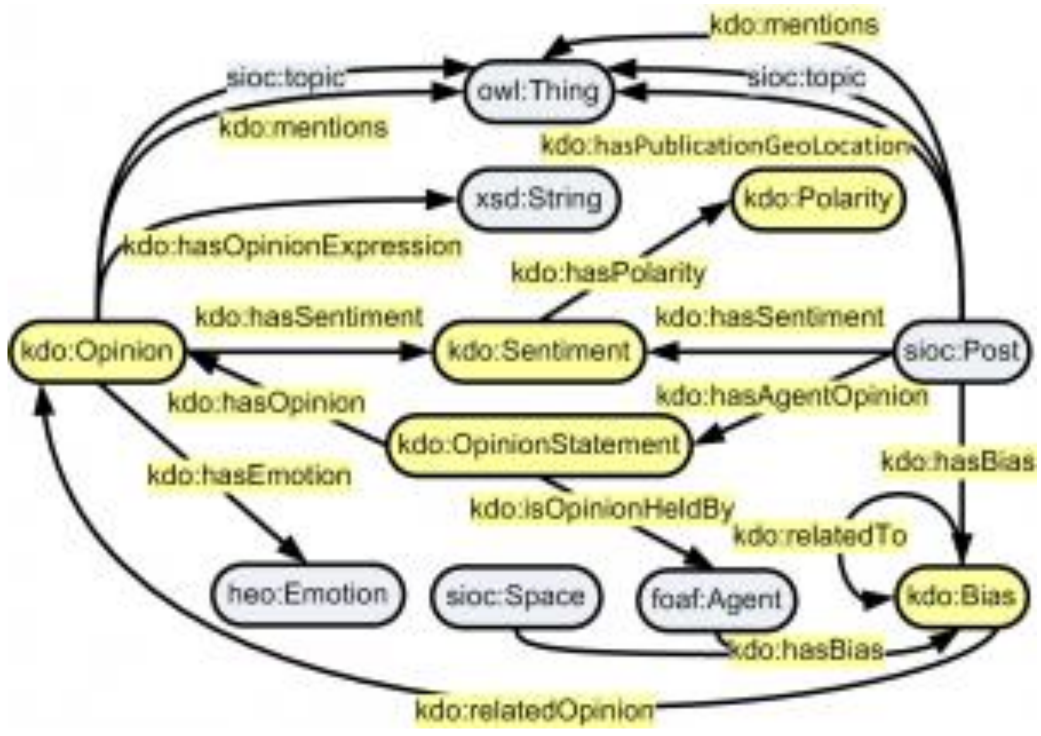


Fig. 5.12. Knowledge Diversity Ontology (http://render-project.eu/resources/kdo/).

A formalization of this methodology is presented in next sub sections by means of a conceptual sociolinguistic dialogue model for understanding how learning evolves and how cognitive process is constructed (Caballé et al 2011).

5.2.1 A dialogue model for modelling and represent collaborative interaction data

The model proposed here is based on the integration of several models and methods: the Negotiation Linguistic Exchange Model (Martin, 1992); a model of Discourse Contributions (Clark and Schaefer, 1989); and the types of learning actions underlying a participant turn (Self, 1994). The structure of a long interaction is constructed cooperatively by using the exchange as the basic unit for communicating knowledge. Following (Martin, 1992), three general exchange structure categories are considered: *give-information* exchange, *elicit-information* exchange and *raise-an-issue* exchange, which consist of different types of moves (Schwartz, 1999) and describe a generic discourse goal.

More specifically, the goal of the actor who initiates the *give-information* exchange is to inform his/her partners about a certain situation with the aim to change the partners' mental states. Informing includes moves that explain, give an opinion, describe or remind a situation in different ways. The actor goal of the *elicit-information* exchange is to elicit the partners' state of mind (knowledge, beliefs, attitude, desire or abilities) of a situation, in which the actor is not aware or certain about. The actor goal of the *raise-an-issue* exchange is to raise an issue (a problem or question) to be resolved by the participants, which causes to explore their state of mind (knowledge, beliefs, etc.).

According to Martin (1992), there is a move that constitutes the "obligatory move" of the exchange, since it either carries or indicates completion of the discourse goal for which the exchange is initiated. According to Clark and Schaefer (1989), each move is seen as a contribution to discourse. This means that in a cooperative conversation, contributions are regarded as collective acts performed by the participants working together, resulting in units of conversation - typically turns (moves) - that aim to make a success of the discourse they compose. Yet, not all moves contribute in the same way toward the successful completion of the exchange. According to Self (1994), some moves have a pure contributing function toward the realization of the obligatory move of the exchange. In fact, without the presence of those moves, the obligatory move cannot be realized; thus, those moves really contribute toward the realization of the obligatory move. Consequently, it is stated that successful realization of the obligatory move conveys evidence of (initial) success of the exchange. In contrast, others moves have a rather supporting function (provide evidence of support) toward the definite completion of the obligatory move and consequently of the exchange. This is the case of the follow-up moves of the three exchanges. Supporting moves are optional, so they may not be realized. In such a case, they convey an implicit support toward the obligatory move, that is, toward the definitive completion of the exchange.

Based on the work of Self (1994), Puntambekar, S. (2006) and Soller (2001), partners are involved in a process of realizing a number of learning actions which lead to the completion of the exchange goal. Each move type captures and controls the evolution of the learning action performed by a participant by setting the expectations of the type of learning actions which has to be realized next by the other participants so that the goal set by the initial move be accomplished.

Both the quantity and the quality of the several move types performed are measured by the collaborative effort of the members involved to achieve the discourse goal of an exchange. The term collaborative effort means both the number of contributing and supporting moves issued by a participant, which indicates an active participation (distinguishing between proactive and reactive one) or passive one. It is also considered the type and effectiveness of these moves, which indicate the way a participant contributes toward the achievement of the shared discourse goal, as regards knowledge possession and transfer, reasoning capability and positive attitude. The tutor measures move effectiveness by assessing the quality of their content. In addition, peer assessment can be effected to complete the evaluation of each contribution made in form of post. The roles that these moves play in the exchange as well as the degree of success of that role determine the successful completion of the exchange goal.

Completion of an exchange expresses the mutual beliefs of all participants about the accomplishment of its discourse goal. Moreover, it implies the achievement of a certain degree of knowledge building and distribution among the different participants. This degree can be deduced and measured by exploring the principal interaction indicators proposed by this model. For each participant the model measures: the total number of moves created, his/her participation behavior (proactive, reactive, supportive, or passive), the effectiveness and impact that each move has in the discourse and in the achievement of the current discourse goal, as well as the evaluation of the move content and significance by his/her peers and the tutor.

5.2.2 Representing collaborative interaction data at different description levels

In general, the three general types of exchanges presented represent standard discourse structures for handling information and suggest a certain type of knowledge building, as a result of giving and eliciting information or working out a solution on an issue set up. These discursive structures enable the participants to take turns, share information, exchange views, monitor the work done and plan ahead. Most importantly, they provide a means to represent and operationalize the cognitive product at individual level, that is, the way the reasoning process is distributed over the participants as it is shared in a collaborative discourse.

Consequently, interaction analysis takes into account both the way the interaction is structured and the types of contributions or posts, which are represented by the ontological approach presented in the previous section and particularized in Table 5.3. The analysis results yield very useful conclusions on aspects such as individual and group working, dynamics, performance and success, which allows the tutor to obtain a global account of the progress of the individual and group work and thus to identify possible conflicts and monitor the whole learning process much better.

Table 5.3: Exchange moves taken into account and their categories.

Exchange moves	**Categories**
support	Greeting
	Encouragement
	Motivation
elicit-information	REQUEST-Information
	REQUEST -Elaboration
	REQUEST -Clarification
	REQUEST -Justification
	REQUEST-Opinion
	REQUEST-Illustration
give-information	INFORM-Extend

	INFORM-Lead
	INFORM-Suggest
	INFORM-Elaboration
	INFORM-Explain/Clarification
	INFORM-Justify
	INFORM-State
	INFORM-Agree
	INFORM-Disagree
set-up-an-issue	PROBLEM-Statement
provide-solution	PROBLEM-Solution
consent-solution	PROBLEM-Extend solution
	PROBLEM-Assent solution

A further innovation of this model is that it allows participants to end up an exchange, which took several moves to conclude by "replaying" the main contributing move of the exchange. For instance, in a *set-up-an-issue* exchange, a solution move may not be sufficiently complete and thus has to be further elaborated, corrected or extended. To that end, another participant has the option to provide an *extend-solution* move, which completes the initial solution. In general, a "replay" move can be used to resume all the changes produced from the initial appearance of an exchange goal to be achieved to its final conclusion and acceptance by all participants. This can be useful both to reinforce the fact that the goal of the exchange has been completed successfully and to explicitly indicate the progress achieved in the participants' process of knowledge building (especially as regards the participant who provided the main contributing move of the exchange).

Finally, the participant is required to commit certain action to indicate s/he has read a certain post, such as send a reply and assent the contribution. The aim is both to provide reliable indicators on the number of posts read and to promote the discussion's dynamics by increasing the users' interaction with the system.

In overall, our model annotates and examines a variety of elements that contribute to the understanding of the nature of the collaborative interactions, such as the students' passivity, proactivity, reactivity as well as the effectiveness and impact of their contributions to the overall goal of the discussion. The aim is to provide both a deeper understanding of the actual discussion process and a more objective assessment of individual and group activity.

5.3 Experimentation and Validation of CS^2

This section presents an experimental approach to evaluate the ontological framework presented previously in terms of completeness and usefulness by addressing the requirements of a newly created Virtualized Collaborative Session (VCS) system that enables the virtualization of collaborative sessions (Caballe, Gañan et al 2011). The realization of this system is first reported from the requirements that conducted the development of a VCS prototype where our CS^2 ontology is embedded and populated with data that models and represents knowledge coming from live collaborative sessions of different Web-based forums. The specification of the CS^2 data in CSML format inputs the VCS system in order to proceed with the virtualization process. An experiment in a real context of learning is then reported for validation purposes by showing a real collaborative activity supported by the VCS system.

Therefore, the validation of the completeness and usefulness of the CS^2 ontology is achieved by proving that our ontological approach allows for representing the relevant information about collaboration sessions underlying the content of live discussions in any Web-based forum and that it provides all the necessary information for creating virtual collaborative sessions.

5.3.1 Realization of the VCS System

We provide next the main guidelines for the realization of a VCS system (Fig. 5.13). The main feature of a VCS system is to be compatible with different kinds of chats, forums and collaborative sessions in general. For the sake of our experiments we used two very different Web forums: the Discussion Forum (DF) and the Intelligent Web Teacher (IWT). As an input of the VCS system, we used an XML file containing the collaborative session data in the CSML common format. The CSML specifies the knowledge of collaborative sessions from both web-based forums.

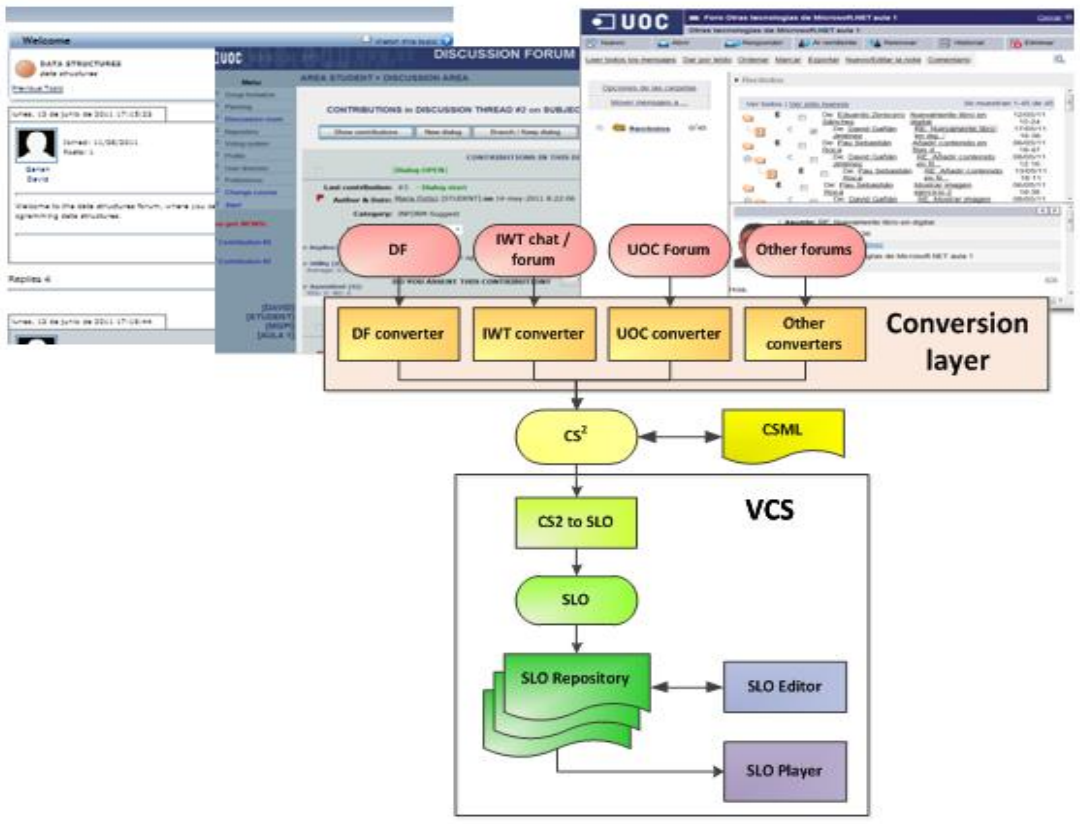
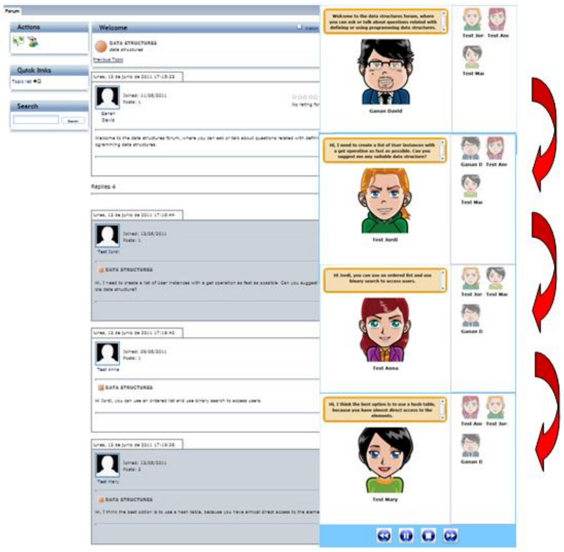
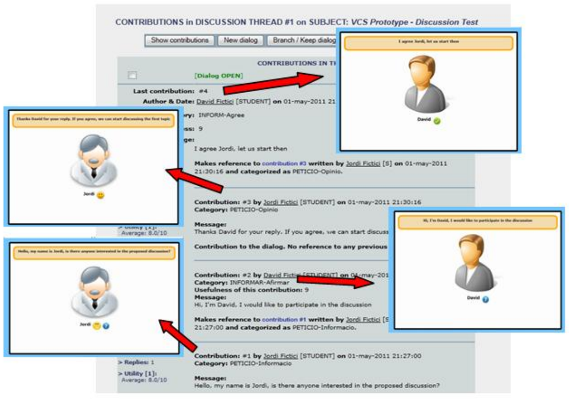


Fig. 5.13. Architecture of the VCS system, which is compatible with multiple forums and chats by specific converters.

The process of conversion between the two sources of collaborative session data and CSML was done by developing specific converters (see Fig. 5.13), which were different for each kind of source (i.e., the data models of both IWT and DF forums). Then, the VCS system processed data in CSML format and created a complex learning object named Storyboard Learning Object (SLO), containing information about scenes, characters, and other artefacts used during the later visualization of this learning object. This information could be edited and played in a multimedia fashion in order to enable moderators and learners to observe the virtualized collaborative session in an interactive way.


Forum
Actions
Quick links
Search
Welcome
DATA STRUCTURES
Replies 4
Welcome to the data structures forum, where you can ask or talk about questions related with defining or using programming data structures.
Ganan David
Hi, I need to create a list of User instances with a get operation as fast as possible. Can you suggest me any suitable data structure?
Test Jordi
Hi Jordi, you can use an ordered list and use binary search to access users.
Test Anna
Hi, I think the best option is to use a hash table, because you have almost direct access to the elements.
Test Mary


CONTRIBUTIONS in DISCUSSION THREAD #1 on SUBJECT: *VCS Prototype - Discussion Test*

Show contributions | New dialog | Branch / Keep dialog

[Dialog OPEN]

Last contribution: #4
Author & Date: David Fictici [STUDENT] on 01-may-2011 21
ry: INFORM-Agree
ss: 9
ge:
I agree Jordi, let us start then
Makes reference to contribution #3 written by Jordi Fictici [S] on 01-may-2011 21:30:16 and categorized as PETICIO-Opinio.

Contribution: #3 by Jordi Fictici [STUDENT] on 01-may-2011 21:30:16
Category: PETICIO-Opinio
Message:
Thanks David for your reply. If you agree, we can start discuss
Contribution to the dialog. No reference to any previous

Average: 8.0/10

Contribution: #2 by David Fictici [STUDENT] on 01-may-201
Category: INFORMAR-Afirmar
Usefulness of this contribution: 9
Message:
Hi, I'm David, I would like to participate in the discussion
Makes reference to contribution #1 written by Jordi Fictici [S
21:27:00 and categorized as PETICIO-Informacio.

> Replies: 1
> Utility [1]:
Average: 8.0/10

Contribution: #1 by Jordi Fictici [STUDENT] on 01-may-2011 21:27:00
Category: PETICIO-Informacio
Message:
Hello, my name is Jordi, is there anyone interested in the proposed discussion?

I agree Jordi, let us start then
David
Thanks David for your reply. If you agree, we can start discussing the first topic
Jordi
Hi, I'm David, I would like to participate in the discussion
David
Hello, my name is Jordi, is there anyone interested in the proposed discussion?
Jordi

Fig.5.14. Samples of sequence of storyboard scenes from the VCS prototype with a discussion evolving over time after the virtualization of two different live collaborative sessions performed in the IWT and DF forums.

Overall, the VCS transformed live discussion sessions into animated storyboards consumed by learners, sessions evolved ("animate") over time, and the ultimate end-user interactions were handled. As a result, the VCS provided an attractive learning resource so that learners became more motivated and engaged in the collaborative activities (see Fig. 5.14).

5.3.2 Experience in a real learning context

The real context of this experience is the virtual learning environment of the Open University of Catalonia (UOC)[56]. Given the added value of asynchronous discussion groups, the UOC have incorporated on-line discussions as one of the pillars of its pedagogical model. To this end, great efforts are being made to develop adequate on-line tools to support the essential aspects of the discussion process, which include students' monitoring and evaluation as well as engagement in the collaboration.

In order to evaluate the prototype of the VCS and analyze its effects in the discussion process, the sample of the experiment consisted of 81 graduated students enrolled in the course Organization

[56] The Open University of Catalonia (UOC) is located in Barcelona, Spain. Since 1995, UOC offers full on-line distance tertiary education via Internet to currently 60,000 students. http://www.uoc.edu

Management and Computer Science Projects from the Computer Science degree at the UOC were involved in this experience. Students were equally distributed into two classrooms and participated in the experience at the same time. Students from each classroom were required to use standard text-based discussion forums to support the same discussion with the same rules during the same time. In addition, in one of the classrooms (experimental group) the standard forum was equipped with the multimedia-based VCS tool. In the other classroom (control group) the VCS was not available.

The in-class collaborative assignment in both groups lasted three weeks in the Fall term and consisted of discussing the same issue: “Factors that lead a Computer Science project to failure”. In this assignment, each student was required to post one contribution at least on the issue in hand. During the discussion, any student could contribute as many times as needed in the discussion forum by posting new contribution, replying to others as well as start extra discussion threads to provide new argumentations with regards to the issue addressed. In addition, in one classroom, participants could follow the discussion also by the VCS. The aim was to evaluate the effects of the VCS system in the participation by comparing the activity levels of the discussion between the two groups.

5.3.3 Data Elaboration and Interpretation of the Results

The data from this experience was collected by means of the web-based forums supporting the discussions in each classroom. Moreover, specific data from the interaction with the VCS system was also collected considering the following validation criteria (Caballe, Gañan et al 2011):

- Evaluate the level of fulfilment of the VCS tool features.
- Evaluate the level of participation of students with the inclusion of the VCS in the discussion.

All quantitative data collected was stored in databases and log files considering the following quantitative metrics:

- Number of VCS created with the VCS tool.
- Number of students using the VCS.
- Number of messages submitted by students when the VCS is used.
- Number of messages submitted by students when no VCS is used.
- Number of views when the VCS is used.
- Number of views when no VCS is used.
- Number of words written by students when the VCS is used.
- Number of words written by students when no VCS is used.

Table 5.4: summary of the activity levels of the discussion in both control and experimental groups.

Metrics Statistics	Standard Forum (control)	Standard Forum + VCS (experimental)
Number of students	40	41
Total of posts	119	151
Mean posts/student	M=2.9	M=3.6
Total words	33942	27091
Mean words/post	M=285	M=179
Total views	2149	1889
Mean views/student	M=18.0	M=12.5

Analyzing the results of table 5.4, it seems that by using the VCS the participation quantity is fostered since the number of posts is higher. On the other hand, the number of views (i.e., readings) of text posts are lower in the forum that the VCS has, pointing out that some of the students have seen in the storyboard as an alternative to text posts, which is confirmed by the data collected from the activity logs of the VCS.

Finally, participation quality is shown in terms of the number of words per post. The lower mean statistics of words per post in the experimental group may mean that the users of the VCS were more effective and dynamic when communicating their ideas by either new posts or reply posts. As a result, the contributions became more structured and specific whereas the control group promoted larger monolithic one-sided points of view.

5.3.4 Validation

The validity of the CS^2 ontology has been tested on three levels, namely correctness, completeness and usefulness.

First, the correctness of the ontology has been verified using the reasoners available within Protégé[57] over CSML, thus allowing for determining that the ontology is well written on a formal level, which means that it contains no contradictions and therefore it can be instantiated.

Second, the ontology completeness has been validated naively by the experiment presented in the previous section, showing how the ontology has been used to represent information of several real forums of virtual environments related to computer science subjects. As we have demonstrated with this experiment, the current CS^2 specification allows for representing the relevant information about collaboration sessions underlying the content of IWT and DF Forums. Hence, since the ontology has been able to deal with the relevant information of the forums we can state that it allows for representing the information of the domain we are interested in. Nevertheless, the completeness of the ontology cannot be validated formally as it is an ontology for open environments (Gómez-Pérez, Fernández-López, Corcho 2004).

Finally, the usefulness of the ontology has been proved by a naive validation of participation in a

[57] "The Protegé Ontology Editor and Knowledge Acquisition System" retrieved from http://protege.stanford.edu

collaborative learning activity supported by the VCS (see (Caballe, Gañan et al 2011) for a full validation of the VCS from other relevant dimensions, such as learner's engagement and acquisition of knowledge). The results show higher level of activity in the forum tool equipped with the VCS in comparison to the standard forum tool without the VCS.

5.4 Chapter summary

In this chapter all theoretical approaches presented in previous chapters were put together and motivated a new ontology called Collaborative Session Conceptual Schema based on SIOC and FOAF for modeling knowledge from online collaborative sessions within Web forums. We described the knowledge this ontology needs to have and the ontologies we use in order to represent such knowledge. Then, we defined the schema structure of our ontology that allow for storing and working with collaborative session data and the relationships between the different classes of the ontology. In order to specify the knowledge modeled by this ontology we proposed a single-purposed ontology language named Collaborative Session Markup Language based on RDF. Thereafter, we have extended the ontology to deal with the emotions and sentiments the students have when collaborating. Such extension has been done using the Knowledge Diversity Ontology and the Human Emotions Ontology. The data collected in our ontology is to be afterwards transformed later on into useful knowledge about what is happening during the collaboration. To make it possible this step, a methodology based on a dialogue is proposed to model and represent specific collaborative interaction data from our ontology. This model is based on primitive exchange moves found in any forum posts, which are then categorized at different description levels with the aim to effectively collect and classify the type and intention of forum posts. Finally, an experiment has been conducted in order to validate the created ontology.

References

Alani H. and Brewster C. 2006. Metrics for ranking ontologies. 4th Int'l. EON workshop, 15th Int'l world wide web confCiteseer.

Babic F., Wagner J. and Paralic J. 2008. The role of ontologies in collaborative systems. Applied machine intelligence and informatics, 2008. SAMI 2008. 6th international symposium onIEEE. 119 p.

Bagüés MI, Bermúdez J, Illarramendi A, Tablado A, Goñi A. 2003. Using ontologies in the development of an innovating system for elderly people tele-assistance. In: Proceedings of on the move to meaningful internet systems 2003: CoopIS, DOA, and ODBASE. .

Barros B, Mizoguchi R, Verdejo M. A platform for collaboration analysis in CSCL. an ontological approach.

Barros B, Verdejo M, Read T, Mizoguchi R. 2002. Applications of a collaborative learning ontology. MICAI 2002: Advances in Artificial Intelligence :103-18.

Bechhofer S, Harmelen Fv, Hendler J, Horrocks I, McGuinness DL, Patel-Schneider PF, Stein LA. December, 2003. OWL web ontology language reference , http://www.w3.org/TR/owl-ref/. W3C.

Berlanga A and Garcia FJ. 2005. Learning technology specifications: Semantic objects for adaptive learning environments. International Journal of Learning Technology 1(4):458-72.

Bernaras A, Laresgoiti I, Corera J. 1996. Building and reusing ontologies for electrical network applications. In: Proceedings of the twelfth european conference on artificial intelligence (ECAI). Budapest, Hungary.

Board IS. 1998. IEEE recommended practice for software requirements specifications. IEEE Computer Society.

Bodenreider O. 2004. The unified medical language system (UMLS): Integrating biomedical terminology. Nucleic Acids Res 1(32).

Borst WN. 1997. Construction of engineering ontologies for knowledge sharing and reuse. University of Tweenty.

Brank J., Grobelnik M. and Mladenic D. 2005. A survey of ontology evaluation techniques. Proceedings of the conference on data mining and data warehouses (SiKDD 2005)Citeseer.

Bray T, Paoli J, Sperberg-MCQueen CM, Maler E, Yergeau F. February 2004. Extensible markup language (XML) recommendation http://www.w3.org/TR/REC-xml/. W3C.

Brickley D and Guha RV. December, 2003. {RDF vocabulary description language 1.0: RDF schema}, \refurl{http://www.w3.org/TR/rdf-schema/}. W3C.

Burton-Jones A, Storey VC, Sugumaran V, Ahluwalia P. 2005. A semiotic metrics suite for assessing the quality of ontologies. Data Knowl Eng 55(1):84-102.

Caballé, S., Daradoumis, T., Xhafa X., Juan, A. (2011). Providing Effective Feedback, Monitoring and Evaluation to On-line Collaborative Learning Discussions. Computers in Human Behavior, 27(4), 1372-1381, Elsevier.

Caballé, S., Daradoumis, T., Xhafa, F., Conesa, J. (2010). Enhancing Knowledge Management in Online Collaborative Learning. International Journal of Software Engineering and Knowledge Engineering (IJSEKE). Vol. 20, No. 4, pp. 485-497. World Scientific.

Caballé, S., Gañan, D. Dunwell, I., Pierri, A., Daradoumis, T. (2011). CC-LO: Embedding Interactivity, Challenge and Empowerment into Collaborative Learning Sessions. Journal of Universal Computer Science. In press. Retrieved from http://clpl.uoc.edu/JUCS_CaballeEtAl

Calvanese D, Lenzerini M, Nardi D. 1998. Description logics for conceptual data modeling. In: Logics for databases and information systems. Kluwer. 229 p.

Chandrasekaran B, Josephson JR, Benjamins VR. 1999. What are ontologies, and why do we need them? IEEE Inteligent Systems 14(1):20--26.

Ciancarini Paolo and Presutti Valentina. 2002. Towards ontology driven software design. Radical innovations of software and systems engineering in the future: Nineteenth international workshop, RISSEF 2002; October 2002; Venice, Italy. Springer-Verlag. 122 p.

Clark, H. & Schaefer, E. (1989). Contributing to discourse. Cognitive Science. 13(2), 259-294.

Conen W and Klapsing R. 2000. A logical interpretation of RDF. Journal of Electronic Transactions on Artificial Intelligence (ETAI), Area: The Semantic Web (SEWEB 5.

Conesa J, Storey VC, Sugumaran V. 2010. Usability of upper level ontologies: The case of ResearchCyc. Data Knowl Eng 69(4):343-56.

Conesa J. 2008. Ontology driven information systems development: Pruning and refactoring of ontologies. PhD thesis Barcelona: UPC.

Conesa J, Palol Xd, Olivé A. 2003. Building conceptual schemas by refining general ontologies. In: In proceedings of the fourteenth international conference on database and expert systems applications. . 693 p.

Connolly D, Harmelen Fv, Horrocks I, McGuinness DL, Patel-Schneider PF, Stein LA. March, 2001. DAML+OIL reference description, http://www.w3.org/TR/daml+oil-reference. W3C.

Corinna, C.; Vapnik, V. (1995). Support-Vector Networks. Machine Learning, 20.

Cristani M and Cuel R. 2004. A comprehensive guideline for building a domain ontology from scratch. In: Proceedings of the fourth international conference on knowledge management (I-KNOW'04). Graz, Austria: .

Daradoumis, T., Xhafa, F. and Marquès, J.M. (2003) Exploring Interaction Behaviour and Performance of Online Collaborative Learning Teams, 9th Int. Workshop on Groupware, CRIWG'03, France.

Davenport TH and Prusak L. 2000. Working knowledge: How organizations manage what they know. Harvard Business Press.

Deerwester, S.; Dumais, S.; Furnas, G.; Landauer, T.; Harshman, R. (1990). “Indexing by Latent Semantic Analysis”. Journal of the American Society for Information Science 41 (6): 391-407.

Dixon NM. 2000. Common knowledge: How companies thrive by sharing what they know. Harvard Business Press.

Embley DW. 2004. Toward semantic understanding - an approach based on information extraction ontologies. In: Proceedings of the fifteenth conference on australasian database. Dunedin: . 3 p.

Falbo RdA, Guizzardi G, Duarte KC. 2002. An ontological approach to domain engineering. In: Proceedings of the fourteenth international conference on software engineering and knowledge engineering. . 351 p.

Farquhar A, Fikes R, Rice J. 1997. The ontolingua server: A tool for collaborative ontology construction. Int J Hum -Comput Stud 46(6):707-27.

Fensel D. 2004. Ontologies: A silver bullet for knowledge management and electronic commerce. Springer Verlag.

Fensel D, Lausen H, Polleres A, de Bruijn J, Stollberg M, Roman D, Domingue J. 2007. Enabling semantic web services. Springer-Verlag Berlin Heidelberg.

Fensel D, Horrocks I, Harmelen Fv, Decker S, Erdmann M, Klein M. 2000. {OIL} in a nutshell. In: Proc. of the 12th european workshop on knowledge acquisition, modeling, and management ({EKAW}'00). Dieng R, editor. Springer-Verlag. 1 p.

Gangemi A., Catenacci C., Ciaramita M. and Lehmann J. 2005. A theoretical framework for ontology evaluation and validation. Semantic web applications and perspectives (SWAP)–2nd italian semantic web workshopCiteseer.

Genesereth MR, Fikes RE, Computer Science Department, Stanford University. 1992. Knowledge interchange format-version 3.0: Reference manual. Citeseer.

Genesereth MR and Nilson NJ. 1987. Logical foundation of artificial intelligence. Los Altos (California):

Gómez-Pérez A, Fernández-López M, Corcho O. 2004. Ontological engineering: With examples from the areas of knowledge management, e-commerce and the semantic web. Springer Verlag.

Gómez-Perez A, Fernandez-López M, Corcho O. 2004. Ontological engineering. Springer.

Good B. M., Tranfield E. M., Tan P. C., Shehata M., Singhera G. K., Gosselink J., Okon E. B. and Wilkinson M. D. 2006. Fast, cheap and out of control: A zero curation model for ontology development. Pacific symposium on biocomputingCiteseer. 128–139 p.

Grant RM. 1996. Prospering in dynamically-competitive environments: Organizational capability as knowledge integration. Organization Science 7(4):375-87.

Gruber TR. 2004. Every ontology is a treaty - a social agreement - among people with some common motive in sharing (tomas R. gruber's interview). SiG SEMIS - Semantic Web and Information Systems 1(3):4-8.

Gruber TR. 1993a. Toward principles for the design of ontologies used for knowledge sharing. In: Formal ontology in conceptual analysis and knowledge representation. Guarino N and Poli R, editors. Deventer, The Netherlands: Kluwer Academic Publishers.

Gruber TR. 1993b. A translation approach to portable ontology specifications. Knowledge Acquisition 5(2):199-220.

Guarino N. 1998. Formal ontology and information systems. In: Proceedings of the FOIS'98. IOS Press. 3 p.

Guarino N and Welty C. 2002. Evaluating ontological decisions with ontoclean. Communications of the ACM 45(2):61-5.

Guarino N and Giaretta P. 1995. Ontologies and knowledge bases. towards a terminological clarification. In: Towards very large knowledge bases: Knowledge building and knowledge sharing. Mars N, editor. Amsterdam: . 25 p.

Guarino N, Masolo C, Vetere G. 1999. OntoSeek: Content-based access to the web. IEEE Intelligent Systems 14(3):70-80.

Hartmann J, Spyns P, Giboin A, Maynard D, Cuel R, Suarez-Figueroa MC, Sure Y. 2004. Methods for ontology evaluation. Knowledge Web Deliverable D 2.

Hayes P. 2001. Rdf model theory. .

Heijst Gv, Spek Rvd, Wielinga BJ. 1997. Using explicit ontologies in KBS development. International Journal of Human-Computer Studies 45:183-292.

Hepp M, De Leenheer P, De Moor A. 2007. Ontology management: Semantic web, semantic web services, and business applications. Springer-Verlag New York Inc.

Herman I. 2007. Introduction to the semantic web. International conference on dublin core and metadata applications.

Holsapple CW and Joshi KD. 2002. A collaborative approach to ontology design. Communications of the ACM 45(2):42-7.

Horrocks I, Patel-Schneider PF, Boley H, Tabet S, Grosof B, Dean M. 2004. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission 21.

Inaba A., Supnithi T., Ikeda M., Mizoguchi R. and Toyoda J. 2000. An overview of" learning goal ontology. Proceedings of the workshop analysis and modelling of collaborative learning interactions at the european conference on artificial intelligence ECAI-2000. berlin, germany.

Karp PD, Chaudhri VK, Thomere J. 1999. XOL: An XML-based ontology exchange language. Version 0.3, July 3.

Karp PD. 2000. An ontology for biological function based on molecular interactions. Bioinformatics 16(3):269-85.

Khoshafian SN and Copeland GP. 1986. Object identity. ACM SIGPLAN Notices 21(11):406-16.

Kifer M. 2005. Rules and ontologies in f-logic. Reasoning Web :22-34.

Kifer M., De Bruijn J., Boley H. and Fensel D. 2010. A realistic architecture for the semantic web. Proceedings of the international conference on rules and rule markup languages for the semantic web (RuleML 2005). 17 p.

Kifer M, Lausen G, Wu J. 1995. Logical foundations of object-oriented and frame-based languages. Journal of the ACM 42(4):741--843.

Kim HM and Sengupta A. 2007. Extracting knowledge from XML document repository: A semantic web-based approach. Information Technology and Management 8(3):205-21.

Kishore R, Zhang H, Ramesh R. 2004. A HELIX-SPINDLE MODEL for ontological engineering. Communications of the ACM 47(2):69-75.

Klyne G, Carroll JJ, McBride B. 2004. Resource description framework (RDF): Concepts and abstract syntax. Changes .

Knight K and Luk S. 1994. Building a large knowledge base for machine translation. In: American association of artificial intelligence conference (AAAI'94). .

Lassila O and McGuinness D. 2001. The role of frame-based representation on the semantic web. Stanford, California: Stanford University. Report nr KSL-01-02. 9 p. .

Lee TB, Hendler J, Lassila O. 2001. The semantic web. Sci Am 284(5):34-43.

Lenat DB. 1995. A large-scale investment in knowledge infrastructure. Communications of the ACM 38:32.38.

Lenat DB, Guha RV, Pittman K, Pratt D, Shepherd M. 1990. CYC: Toward programs with common sense. Communications of the ACM 33(8):30-49.

Leshed, G.; Kaye, J. (2006). "Understanding how bloggers feel: recognizing affect in blog posts". CHI extended abstracts (pp. 1019-1024), ACM.

Li M, Wang D, Du X, Wang S. 2005. Ontology construction for semantic web: A role-based collaborative development method. Web Technologies Research and Development-APWeb 2005 :609-19.

Lindvall M and Rus I. 2002. Knowledge management in software engineering. IEEE Software 19(3):26-38.

Locopoulos P. 1992. Conceptual modeling. In: Conceptual modeling, databases and CASE: An integrated view of information systems development. Locopoulos P and Zicari R, editors. Wiley. 1 p.

Luke S and Heflin J. April, 2000. SHOE 1.01 proposed specification,\\ \refurl{http://www.cs.umd.edu/projects/plus/SHOE/spec.html}. SHOE Project.

MacGregor RM. 1991. ACM SIGART 2(3):88--92.

Martin, J.R. (1992). English Text: Systems and Structure. Amsterdam: Benjamin Press.

Maedche A and Staab S. 2001. Ontology learning for the semantic web. IEEE Inteligent Systems 16(1):72-9.

Mahesh K, Nirenburg S, Cowie J, Farwell D. 1996. An assessment of cyc for natural language processing. Las Cruces, New Mexico: New Mexico State University. Report nr MCCS-96-302.

Marco Grassi. 2009. Developing HEO human emotions ontology. In Proceedings of the 2009 joint COST 2101 and 2102 international conference on Biometric ID management and multimodal communication (BioID_MultiComm'09), Julian Fierrez, Javier Ortega-Garcia, Anna Esposito, Andrzej Drygajlo, and Marcos Faundez-Zanuy (Eds.). Springer-Verlag, Berlin, Heidelberg, 244-251.

Milton S, Kazmierczak E, Keen C. 2002. On the study of data modelling languages using chisholm's ontology. In: Information modelling and knowledge bases XIII. Kangassalo H, Jaakkola H, Kawaguchi E, and others, editors. Amsterdam: . 13 p.

Mizoguchi R, Vanwelkenhuysen J, Ikeda M. 1995. Task ontology for reuse of problem solving knowledge. In: Proceedings of towards very large knowledge bases: Knowledge building and knowledge sharing (KBKS'95). Amsterdam: IOS Press. 46 p.

Motta E. 1999. Reusable components for knowledge modelling: Case studies in parametric design problem solving. Ios Pr Inc.

Neches R, Fikes R, Finin T, Gruber T, Patil R, Senator T, Swartout WR. 1991. Enabling technology for knowledge sharing. AI Magazine 12(3):36-56.

Nidumolu SR, Subramani M, Aldrich A. 2001. Situated learning and the situated knowledge web: Exploring the ground beneath knowledge management. J Manage Inf Syst 18(1):115-50.

Noy NF. 2004. Semantic integration: A survey of ontology-based approaches. ACM Sigmod Record 33(4):65-70.

Noy NF and McGuinness DL. 2001. Ontology development 101: A guide to creating your first ontology. Stanford: Stanford Knowledge Systems Laboratory. Report nr KSL-01-05.

Obrst L, Cassidy P, Ray S, Smith B, Soergel D, West M, Yim P. 2006. The 2006 upper ontology summit joint communiqué. Applied Ontology 1(2):203-11.

Olivé A. 2007. Conceptual modeling of information systems. .

Olivé A. 2004. On the role of conceptual schemas in information systems development. In: Proceedings of reliable software technologies - ada-europe. Albert Llamosí AS, editor. Palma de Mallorca (Spain): .

OMG. August, 2003. UML 2.0 superstructure specification, 2.0 edition. OMG.

Patel-Schneider P and Fensel D. 2002. Layering the semantic web: Problems and directions. The Semantic Web—ISWC 2002 :16-29.

Pazzaglia JR and Embury SM. 1998. Bottom-up integration of ontologies in a database context. In: KRDB'98 workshop on innovative application programming and query interfaces, Seattle, USA: .

Peterson BJ, Andersen WA, Engel J. 1998. Knowledge bus: Generating application-focused databases from large ontologies. In: KRDB'98. . 2.1 p.

Pisanelli DM, Gangemi A, Steve G. 2002. Ontologies and information systems: The marriage of the century? In: Proceedings of lyee workshop. Paris: .

Poli R. 2002. Descriptive, formal and formalized ontologies. International Journal of Human-Computer Studies 56(6):639-64.

Puntambekar, S. (2006). Analyzing collaborative interactions: divergence, shared understanding and construction of knowledge. Computers & Education. 47(3): 332-351, Academic Press: Elsevier Ltd.

Ras ZW and Dardzinska A. 2004. Ontology-based distributed autonomous knowledge systems. Information Systems 29:47-58.

Read, J. (2004). "Recognising affect in text using pointwise-mutual information". University of Sussex.

Softic, S. and Hausenblas, M. 2008. Towards opinion mining through tracing discussions on the web. Proceedings of the Social Data on the Web (SDoW 2008) Workshop at the 7th International Semantic Web Conference, Karlsruhe, Germany.

Sabou M, Lopez V, Motta E. 2006. Ontology selection for the real semantic web: How to cover the Queen's birthday dinner? Managing Knowledge in a World of Networks :96-111.

Schreiber G, Wielinga BJ, Jansweijer WNH. 1995. The kactus view on the O'word. In: Workshop on basic ontological issues in knowledge sharing (IJCAI). Montreal, Canada.

Self, J.A. (1994). Dormobilea: vehicle for metacognition. In: Chan, T.W. and Self, J.A. (Eds.), Emerging Computer Technologies in Education, (pp. 1-20). Charlottesville, VA: AACE.

Shanks G, Tansley E, Weber R. 2003. Using ontology to validate conceptual models. Communications of the ACM 46(10):85-9.

Sicilia MA, García-Barriocanal E, Sánchez-Alonso S, Rodríguez-García D. 2009. Ontologies of engineering knowledge: General structure and the case of software engineering. The Knowledge Engineering Review 24(03):309-26.

Smith B. Ontology and information systems, http://wings.buffalo.edu/philosophy/faculty/smith/articles/ontologies.htm.

Soller, A. Supporting Social Interaction in an Intelligent Collaborative Learning System. Int. J. of Artificial Intelligence in Education, 12, (2001) 40-62

Sowa JF. 2000. Knowledge representation: Logical, philosophical and computational foundations. .

Staab S, Erdmann M, Maedche E, Decker S. 2007. ECDL 2000 workshop on the semantic web an extensible approach for modeling. .

Studer R, Benjamins VR, Fensel D. 1998. Knowledge engineering: Principles and methods. IEEE Transactions on Data and Knowledge Engineering 25(1-2):161-97.

Sugumaran V and Storey VC. 2002. Ontologies for conceptual modeling: Their creation, use, and management. Data & Knowledge Engineering 42(3):251-71.

Supekar K. 2004. A peer-review approach for ontology evaluation. 8th int. protege confCiteseer. 77–79 p.

Schwartz, D. L. (1999). The productive agency that drives collaborative learning. In: P. Dillenbourg (Ed.), Collaborative learning: Cognitive and computational approaches. (pp. 197–219). NY: Elsevier Science/Permagon.

Swartout WR, Patil R, Knight K, Russ T. 1996. Toward distributed use of large-scale ontologies. In: Proc. tenth knowledge acquisition for knowledge-based systems workshop, canada.

Tartir S., Arpinar I. B., Moore M., Sheth A. P. and Aleman-Meza B. 2005. OntoQA: Metric-based ontology quality analysis. IEEE workshop on knowledge acquisition from distributed, autonomous, semantically heterogeneous data and knowledge sourcesCiteseer. 45 p.

Thalhammer A, Toma I, Hasan R, Simperl E, Vrandecic D. 2011. How to Represent Knowledge Diversity. International Semantic Web Conference (ISWC) 2011. Poster.

Tijerino Y, Embley D, Lonsdale D, Ding Y, Nagy G. 2005. Towards ontology generation from tables. World Wide Web: Internet and Web Information Systems 8:261-85.

Turney, P. (2002). “Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews”.

Uschold M. 1998. Knowledge level modelling: Concepts and terminology. The Knowledge Engineering Review 13(1):5-29.

Uschold M and Gruninger M. June, 1996. Ontologies: Principles, methods and applications. The Knowledge Engineering Review 11(2):93-136.

Volz R, Studer R, Maedche A, Lauser B. 2003. Pruning-based identification of domain ontologies. Journal of Universal Computer Science 9(6):520-9.

W3C. March, 2003. OWL web ontology language reference. W3C.

Weber R. 2010. Ontological issues in accounting information systems. .

Welty C. 2003. Ontology research. AI Magazine 24:11-2.

Wilson R. 2004. The role of ontologies in teaching and learning. TechWatch Reports .

Wollersheim D and Rahayu W. 2002. Methodology for creating a sample subset of dynamic taxonomy to use in navigating medical text databases. In: Proceedings of the 2002 international symposium on database engineering & applications. . 276 p.

Wouters C, Dillon T, Rahayu W, Chang E, Meersman R. 2004. Ontologies on the MOVE. In: Proceedings of the 9th international conference on database systems for advanced applications. Jeju Island, Korea: . 812 p.

Yair Wand and Weber R. 2004. Reflection: Ontology in information systems. J. Database Management 15(2).