WISDOM

Programma di ricerca (cofinanziato dal MIUR, esercizio 2004)
Ricerca Intelligente su Web basata su Ontologie di Dominio
Web Intelligent Search based on DOMain ontologies

Critical Analysis of the emerging ontology languages and standards

Sonia Bergamaschi, Francesco Guerra, Maurizio Vincini

d1.r1 **9 Giugno 2005**

Sommario

Sommario

Tema	Tema 1: Creazione ed estensione di una ontologia di dominio
Codice	d1.r1
Data	9 Giugno 2005
Tipo di prodotto	Rapporto tecnico
Numero di pagine	22
Unità responsabile	MO
Unità coinvolte	MO, TN, BO, RM
Autori	Sonia Bergamaschi, Francesco Guerra, Maurizio Vincini
Autore da contattare	Francesco Guerra, DII-Unimo, via Vignolese 905, 41100 Modena

Critical Analysis of the emerging ontology languages and standards

Sonia Bergamaschi, Francesco Guerra, Maurizio Vincini

9 Giugno 2005

Abstract

Sommario

1 Introduction

In recent years, the development of ontologies has been moving from the realm of Artificial Intelligent laboratories to the desktops of domain experts. Ontologies has become common on the World-Wide Web, where contents have to be exchange (as automatically as possible) among disparate applications and users and consequently the needed of having a precise, formal and explicit semantics raises.

In [27] some reasons to develop ontologies are collected:

- To share common understanding of the structure of information among people or software agents, in order to provide an explicit representation of the contents of the source described by means of the ontology. This scenario, called also *neutral authoring*, may be useful for a company or an organization by developing their own ontology and then developing translators from this ontology to the terminology required by the various target systems;
- To enable reuse of domain knowledge, in order to build representations of large domains by integrating existing ontologies describing portions of the large domain or using a neutral ontology as agreed standard basis for converting/mapping two ontologies;
- To make domain assumption explicit, in order to help new users who must learn what terms in the domain mean, or to facilitate search. An ontology is used as a structuring device for an information repository; this supports the organization and classification of repositories of information at a higher level of abstraction than is commonly used today. They can be used as a sophisticated indexing mechanism into such repositories;
- To separate domain knowledge from the operational knowledge. This kind of ontology is the basis of the "ontology driven Software Engineering"[1] where an ontology of a given domainis used as a basis for specification and development of some software.The idea is to create an ontology that characterizes and specifies the things that the software system must address, and then use this ontology as a (partial) set of requirements for building the software. The benefits of ontology-based specification are best seen when there is a formal link between the ontology and the software;
- To analyze domain knowledge: formal analysis of terms is extremely valuable when both attempting to reuse existing ontologies and extending them.

[1] *http : //www.bpiresearch.com/WP_BPMOntology.pdf*

There are several survey in literature defining what an ontology is and which language may describe it [8, 9, 13, 34, 10, 35, 32, 15, 14, 1]. Starting from these previous works, this technical report collects the main analysis results and tries to define the main features for an ontology language within the WISDOM project. In particular, another language, the ODL_{I^3} language, extended from the standard ODL at the DBgroup from the University of Modena and Reggio Emilia, will be compared to the most common ones and some extension of it will be proposed.

This report proposes in section 2 some definitions of ontology; starting from the found definitions, section 3 tries to define some requirements that an ontology language has to satisfy. Section 4 analyzes the main languages developed to describe ontologies and introduces the ODL_{I^3} language. Finally, we sketch out some conclusions and Appendix A introduces a comparison between the W3C Web Ontology Language OWL and the ODL_{I^3} language.

2 Overview of the ontology definitions

Different research groups expressed different definitions of ontology; many of these contradict one another. In practical terms, an ontology aims at proving a formal explicit description of concepts in a domain of discourse (classes - sometimes called concepts), properties of each concept describing various features and attributes of the concept (slots or roles or properties), and restrictions on slots (facets or role restrictions). Then, developing an ontology includes: defining classes in the ontology, arranging the classes in a taxonomic hierarchy, defining slots and describing allowed values for these slots, arranging the slots in a taxonomic hierarchy and finally filling in values for slots and instances.

In order to give a formal ontology definition, we claim that all the ontology definitions may be synthesized into two possible kinds of "ontology" definition:

1. Taken from Philosophy, where it means a systematic explanation of being
2. Based on the process followed to build the ontologies

These two approaches, that we will analyze with more detail in next section, do not highlight that different research groups think ontologies as "products" having different levels of formalization and with different expressive power. Several papers tried to classify ontologies on the basis of these features, defining "different kinds" of ontologies. Figure 1, taken from [36], shows different kinds of ontologies where the difference among each other is the formalization level. In some cases, the notion of ontology may be diluted including taxonomies as full ontologies [32, 37]. Internet may be one of the cause of the large diffusion of taxonomies: its pervasiveness enabled the growth of standards describing services and goods. These standards, used especially for e-commerce (e.g. UNSPSC, ecl@ss, Naics) in order to describe services and goods provided by companies, provide a consensual conceptualization of a given domain and may be thought of as domain ontologies.

2.1 Definitions from Philosophy

The most quoted definition in literature was written by Gruber [14] where an ontology is *an explicit specification of a conceptualization* Based on Gruber's definition, many definitions of what an ontology is are proposed. In particular, in [32], Studer et al. propose a definition where *Conceptualization refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. Explicit means that the type of concepts used, and the constraints on their use are explicitly defined. Formal refers to the fact that the ontology should be machine-readable. Shared reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group.* Another relevant definition is in [15], where Guarino says *an ontology is a logical theory for*

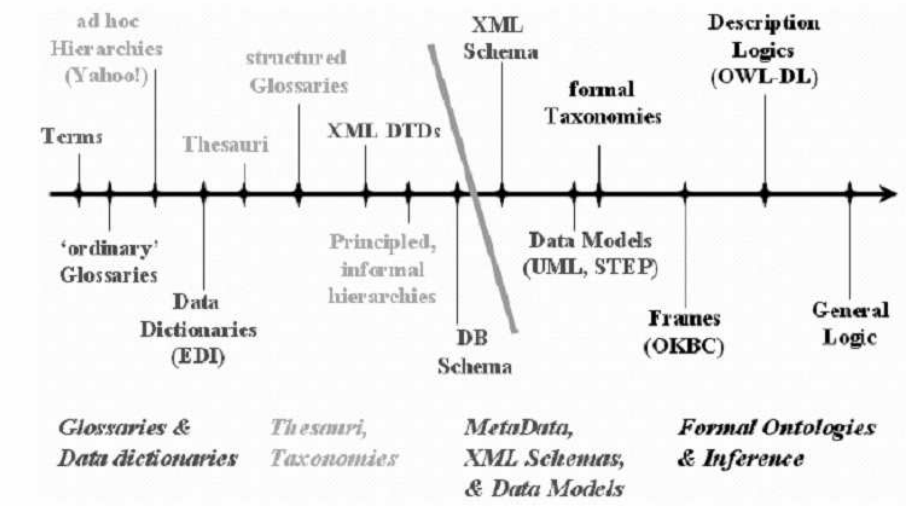

Figure 1: Different kinds of ontologies

the intended meaning of a formal vocabulary, i.e. the ontological commitment to a particular conceptualization of the world.

2.2 Operative definitions

Two are the most relevant definitions found in projects aiming at supporting users in creating and editing ontologies:

- *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* - SENSUS European project [http://www.sensus-int.de/]
- *An ontology is that part of the system which specifies what things exist and what is true about them* - OpenCyc project [http://www.opencyc.org/]

2.3 A general definition

In [14] a large definition of ontology is given: an ontology is a fifth-tuple composed of classes, instances, functions, relationships, axioms: (C,I,R,F,A) where

- C is the set of the concepts, that is the set of the abstractions used to describe the objects of the world;
- I is the set of individuals, that is, the actual objects of the world. The individuals are also called instances of the concept;
- R, is the set of relationships defined on the set C;
- F, is the set of functions defined on the set of concepts and that returns a concept;
- A set of axioms, that is first order logic predicates that constrain the meaning of concepts, relationships and functions.

Analyzing the approaches proposed in literature, we may outline some common points:

- There are different and complementary points of view of the same reality described by the definitions.

- All the definitions presented above highlight a specific aspect of a role played by ontologies.
- All definitions, however, share the idea that an ontology provides a description of a particular viewpoint about a domain and that such a description must be explicit, in that it states a vocabulary for the domain, which is expressed by a certain degree of formality, and that a group commits to use the vocabulary according to the intended meaning associated with it in order to communicate.
- The ontologies may be reused and shared across applications and by groups of people
- An ontology is not independent of the language used to describe it (two ontologies may be different for example in the vocabulary used or in the assumptions while sharing the same conceptualization).

2.4 Ontologies vs conceptual data models

There are many relationships between database schema and ontologies that may be summarized in three categories [36]:

- language expressivity: there is much overlap in expressivity (classes or concepts in ontologies correspond to entities in a E/R model, attributes and relationships in E/R correspond to properties in ontologies languages, ...) and many specific differences in general attributable to the different ways that DB schema and ontologies have been used. Ontologies have a range of purposes including interoperability, search, and software specification. The primary use of most DB schema is to structure a set of instances for querying a single database. This difference impacts heavily on the role of constraints. For ontologies, constraints are called axioms. Their main purpose is to express machine-readable meaning to support accurate automated reasoning. This reasoning can also be used to ensure integrity of instances in a knowledge base. For databases, the primary purpose of constraints is to ensure the integrity of the data (i.e. instances). These "integrity constraints" can also be used to optimize queries and help humans infer the meaning of the terms.
- systems that implement the language: there are some key similarities and differences in systems that implement DB schema languages vs. ontology languages. For both, there are processing engines that can be used to perform reasoning. An important difference is that reasoning over ontologies normally is done by general logic-based theorem provers, specific to the language. Although ontology inference may be used for queries and to ensure integrity of instances, these are optional. The fundamental role of a reasoning engine is to derive new information via automated inference. Inference can also be used to ensure the logical consistency of the ontology itself. Note that deriving new information and checking consistency can take place with or without instances. Classically, such mixing of types with instances does not take place with DB schema and data. This is mainly due to much greater scale and performance requirements for database systems. Another key difference is support for taxonomic reasoning: it is fundamental for nearly all ontology applications, but it is not supported by most DBMS.
- usage scenarios: the different roles of DB schema and ontologies also affect design and other pragmatic issues. For example, there is much effort on normalization for DB schema, with no similarly pervasive analogous step for ontologies. Enforcement of DB integrity constraints is expensive; therefore many constraints identified during modeling are left unstated, resulting in loss of captured meaning.

3 Main features of an ontology language

A complete ontology language would have to satisfy:

1. The language has to express the domain knowledge, i.e. the main static information and knowledge objects about a domain.
2. The language has to provide some inference mechanisms, i.e. how the static structures represented in the domain knowledge can be used to carry out a reasoning process.
3. The language has to include mechanisms to provide the reuse and the integration of previously developed ontologies. Moreover, it has to provide mechanisms for mapping of concepts belonging to different ontologies. The following capabilities have to be supported:
   - Import of other ontologies;
   - Mapping of similar concepts/relations belonging to other ontologies and reconciliation of inconsistencies.
4. The language has to provide mechanisms to manage dynamics. In particular, there are two aspects connected with dynamics:
   - Ontologies may evolve;
   - Ontologies represent an evolving real world.
5. The language has a graphical support (for human readability).
6. Ontologies may be multi-lingual, i.e. the same concept may be described in different languages (for human readability).

3.1 Domain knowledge modelling primitives

With the term *domain knowledge*, we mean the main static information and knowledge objects of a domain. There are several kinds of possible information about a domain. In [9], this information is summarized in the following categories:

- **Concepts/classes/entities**, i.e., in a broad sense, anything about which something is said and then it could be the description of a task, function, action, ... A class may also be considered to be a set of objects which share a common structure and behaviour.
- **Attributes/slots and facets**. Attributes (slots) allows describing and specifing classes. The attributes may be local if belong to a specific concept; instance attribute, if its value may be different for each instance of the concept; class attribute, if the value is the same for all the instances of the concept; polymorph attribute, if the attribute assumes the same name and different behaviour for different concepts. Facets are restrinctions for attributes (type, default type, cardinality constraints, ...).
- **Taxonomies** (of concepts and (in case) of relations). Taxonomies are used to organize the knowledge in the domain using generalization/specialization relationships. implies an extensional knowldge: if A is subclass (subrelation) of B, then every instance of A must also be and instance of B.
- **Relationships** represent interactions between concepts of the domain. In same languages there is no difference between relations and attributes refering an attribute as a relation between a class and a data type.

- **Axioms**, i.e. sentences that are always true. They are included in an ontology in order to constrain its information, verify its correctness and deduct new information.
- **Instances/Individuals/Facts/Claims**, i.e. terms used to represent elements in the domain.
- **Rules** are used to express set of actions and heuristics which can be represented independently from the way they will be used.

3.1.1 Instances modelling

An ontology language may include operators to manage instances. Concerning the representation of a specific domain, ontology languages may provide the following primitives:

- equality or inequality definition: the language may describe two equivalent classes (classes having the same instances) or classes where the instances are different with each other. For example OWL allows defining the following features:
  - equivalentClass: Two classes may be stated to be equivalent. Equivalent classes have the same instances. Equality can be used to create synonymous classes. For example, Car can be stated to be equivalentClass to Automobile. From this a reasoner can deduce that any individual that is an instance of Car is also an instance of Automobile and vice versa.
  - equivalentProperty: Two properties may be stated to be equivalent. Equivalent properties relate one individual to the same set of other individuals. Equality may be used to create synonymous properties.
  - sameAs: Two individuals may be stated to be the same. These constructs may be used to create a number of different names that refer to the same individual. Actually, OWL does not adopt the *unique-names assumption*; just because two instances have a different name or ID does not imply that they are different individuals. To ensure that different equals are recognized as such the user has to explicitely indicate it, by means of the different from operator.
  - differentFrom: An individual may be stated to be different from other individuals. Explicitly stating that individuals are different can be important when using languages such as OWL (and RDF) that do not assume that individuals have one and only one name. For example, with no additional information, a reasoner will not deduce that Frank and Deborah refer to distinct individuals.
  - AllDifferent: A number of individuals may be stated to be mutually distinct in one AllDifferent statement.
- Property restrictions: it allows restrictions to be placed on how properties can be used by instances of a class. In OWL these restrictions may be expressed as follows:
  - allValuesFrom: The restriction allValuesFrom is stated on a property with respect to a class. It means that this property on this particular class has a local range restriction associated with it. Thus if an instance of the class is related by the property to a second individual, then the second individual can be inferred to be an instance of the local range restriction class.
  - someValuesFrom: The restriction someValuesFrom is stated on a property with respect to a class. A particular class may have a restriction on a property that at least one value for that property is of a certain type.

Further knowledge which is between the intensional and extensional level may be modelled: most representative instances of a class and enumeration items for attributes:

- Most representative instances of a class. It may be useful to know the most representative instances when a concepts is general for a specific domain. Starting from our experience in integrating different ontology, we observed that: 1) the name and the description of a concept/class is too general to comunicate the meaning; 2) When a class contains thousand of instances and/or derives from an integration process (e.g. the concept is the synthesys of the same concept in different sources), its name is in general approssimate and does not fit in with the instances. For these reasons, the most significant instances can be useful to know the domain described by the ontology. These significant instances may be found by means of a clustering algorithm. For specific domain[2], a further step may be applied: the main instances may be organized in taxonomies.
- Enumeration items for attributes. These items provide the knowledge about the accepted values for an attribute. These values may be exploited in order to have the knowledge about what really an attribute represents.

3.2 Inference mechanisms

Providing the language with inference mechanisms means to establish how the static structures represented in the domain knowledge can be used to carry out a reasoning process. Figure 2 shows how the domain knowledge may be exploited in order to infer new knowledge.

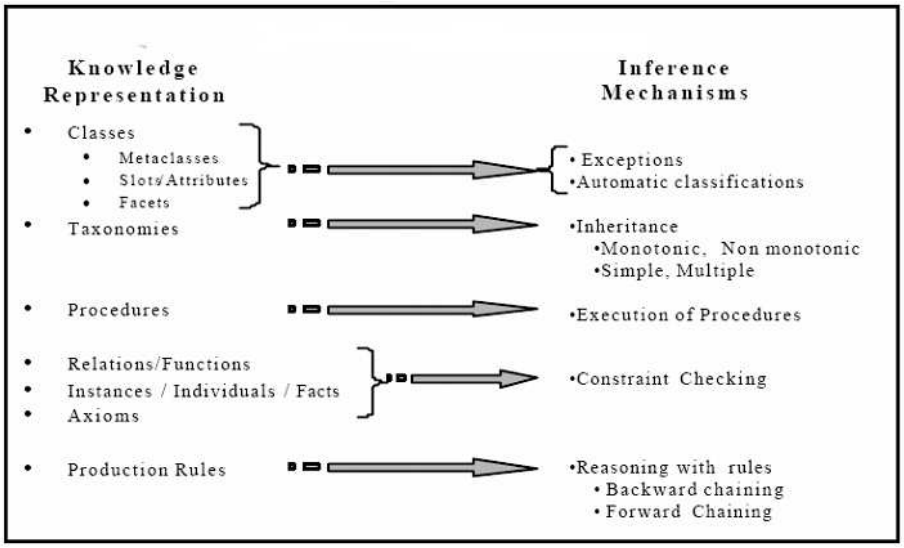

Figure 2: Inference mechanisms used in ontology languages

3.3 Reuse and Integration

The reuse of different ontologies is guaranteed by means the introduction of an "importing" mechanism. By exploiting this operation different ontologies may be reused and inserted as a part of another ontology, allowing the reuse of previously created domain conceptualizations.

[2]We applied a similar process in a e-commerce domain: by analyzing the web sites for enterprises producing mechanic products, we found several ways (more than one thousand) to express the kinds of production. In order to give to the user a more specific knowledge about the enterprises' products, the instances belonging to the "kind of production" were clustered and organized in taxonomy. This process was obtained by exploiting also the taxonomies of products in the original web sites.

The match operation guarantees to identify similar concepts in different ontologies. Results of the matching operations are mapping between terms (also belonging to different ontologies). Ontology languages may define diffent kinds of mapping identifying different types of relationships (similarity, broader term ...). Ontologies integration may generate conflicts that have to be resolved [29]. These conflicts may be generally classified into two types:

- Representation conflicts, i.e. two models describe the same concepts in different way
- Fundamental conflicts, i.e. violations of meta-meta-model (e.g. one type restriction, when the same element is described by means of different data types in different ontologies, cardinality conflicts, ...).

Several papers investigate on mapping discovery [25], i.e. methods and techniques to (semi-) automatically discover and define connections between terms. These methods may exploit a shared ontology (e.g. SUMO [24], DOLCE [11], WordNet[3]) in order to identify similar concepts in different ontology or use heuristic and machine learning techniques. Moreover, some approaches are investigating a declarative formal representations of mapping in order to reason with mapping [25].

In [7], an approach called *contextualizing ontologies* is shown: an ontology is a contextual ontology when its content are kept local, and therefore not shared with other ontologies, and mapped with the contents of other ontologies via explicit context mapping. In this way two levels are defined: one level refers to a general ontology where common concepts are described, the second one allows specifying the common concepts in order to define local representations of them. A particular language, extended from the OWL, was developed in order to represent the map between the common concepts and the local representations.

In [38] and in the technical report D2.R2 "Critical analysis of languages and mapping techniques", the state of the art in ontology mapping is proposed. In [33] there are some ideas for a semantic enrichment for ontology mapping that exploits instance information of the ontology to enrich the original ontology and calculate similarities between concepts in two ontologies.

3.4 Dynamics management

Business dynamics and changes in the operating environment often give rise to continuous changes to application requirements that may be fulfilled only by changing the underlying ontologies. This is especially true for WWW and Semantic Web applications, that are based on heterogeneous and highly distributed information resources and therefore need efficient mechanisms to cope with changes in the environment. So over a period of time an ontology needs to be modified to reflect changes in the real world, changes in user's requirements, drawbacks in the initial design, to incorporate additional functionality or to allow for incremental improvement. In fact very seldom an ontology is perfect the first time it is made, and then continues, without change, to be as useful over time as it was when it was first deployed. The reasons for changes are inherent in the complexity of reality and in the limited ability of humans to cope with this complexity. The changes in ontologies are generated from three possibles events:

- changes in the domain (are comparable to changes in database schemas);
- changes in conceptualization (can result from a changing view of the world and from a change in usage perspective);
- changes in the explicit specification (occur when an ontology is translated from one language into another one).

[3] http://wordnet.princeton.edu/

Ontology development then is a dynamic process starting with an initial rough ontology, which is later revised, refined and filled in the details. Consequently, an ontology almost certainly should be evolved in order:

- to fix "bugs" in the initial design (corrective maintenance);
- to adapt itself to the changes in the environment (adaptative maintenance);
- to improve itself after it has become operational (perfective maintenance);
- to avoid future changes and to alleviate maintenance (preventive maintenance).

The ontology management is becoming more important nowadays. The major reason for this is the increasing number of ontologies in use and the increasing costs associated with adapting them to changing requirements. Developing ontologies and their applications is expensive, but evolving them is even more expensive. However, even though evolution over time is an essential requirement for useful ontologies, appropriate tools and strategies for enabling and managing evolution are still missing. This level of ontology management is necessary not only for the initial development and maintenance of ontologies, but it is essential during deployment, when scalability, availability, reliability and performance are absolutely critical.

To solve this crucial problem several studies have been conducted. In the following we will present two major approach: evolution approach, that try to manage the problem of dynamics in its total complexity and versioning approach, that relies on the use of different version of ontologies to reduce the complication of the problem.

The ontology evolution [23] is the timely adaptation of the ontology as well as the consistent propagation of changes, because a modification in one part of the ontology may generate subtle inconsistencies in other parts of the same ontology, in the instances, depending ontologies and applications. The most important problem to face when a change to an ontology occurs, is to ensure the consistency of the ontology and all the dependent artifacts.

In literature several studies start assuming that the complexity of the dynamics involving ontologies, applications and instances, is too high to maintain everything aligned. A versioning methodology can handle the complexity of the alignment required by a change, providing the user with such a system that manages ontology revisions over time. In a general sense, ontology versioning means that there are multiple variants of an ontology around. In practice, those variants often originate from changes to an existing variant of the ontology and thus form a derivation tree [19]. Then, ontology versioning can be defined as the ability to handle changes in ontologies by creating and managing different variants of it [19]. In order to achieve this ability several methodologies has been studied. In particular methods to distinguish and recognize versions, and procedures for update and change ontologies are developed.

In [26], it is claimed that the traditional distinction between versioning and evolution is not applicable to ontologies: the management of changes is the key issue in the support for evolving ontologies. Hence, they try to combine ontology evolution and versioning into a single concept defined as the ability to manage ontology changes and their effects by creating and maintaining different variants of the ontology. This ability consists of methods to distinguish and recognize versions, specifications of relationships between versions, update and change procedures for ontologies, and access mechanisms that combine different versions of an ontology and the corresponding data.

They distinguish two modes of ontology evolution: traced and untraced evolution.

- *Traced evolution* largely parallels schema-evolution where the evolution is treated as a series of changes in the ontology. After each operation that changes the ontology, the effects on the instance data and related ontologies have to be considered. The resulting effect is determined by the combination of change operations.

- With the *untraced evolution*, two versions of an ontology are proposed and no knowledge of the steps that led from one version to another is given. The problem of finding the differences between (versions of) ontologies is in fact very close to the problem of ontology merging. In both cases, two overlapping ontologies are proposed with the goal to determine a mapping between their elements. When we are merging ontologies, we concentrate on similarities, whereas in evolution we need to highlight the differences. In addition, in the case of ontology evolution we need to make much more "liberal" assumptions in determining which concepts are the same.

The ontology languages do not consider dynamics issues (or they deal with in a trivial way, e.g. see the *priorVersion* tag in OWL. This tag contains three statements able to describe general information about the current version of the ontology `-owl:versionInfo-`, a reference with a backward-compatible ontology `-owl:backwardCompatibleWith-`, and a reference with other ontologies thar are not backward compatible with it).

3.5 Graphical modeling

Several projects tries to use graphical models as semantic networks or UML to define ontologies. In [2], the gap in the expressibility of UML in order to represent ontologies is closed, and in 2003 the OMG proposed a Ontology Definition Metamodel Request For Proposal [28]. Moreover an initial incomplete mapping between the UML and DAML (that can be easily extended to OWL) has been created. In [2] the compatibilities and the incompatibilities between UML and DAML are listed. The more significant difference is related to the DAML concept of property (that has the same meaning and the same structure in OWL and consequently the same remarks are valid in both the languages) is translated into UML. A DAML (UML) ObjectProperty, at a first glance, appears to be the same as a UML association and a DAML (UML) DatatypeProperty appears to be the same as a UML Attribute. This is misleading, since the DAML (UML) notion of ObjectProperty is a first-class modeling element, while UML associations are not. More precisely, an ObjectProperty can exist without specifying any classes that might relate, i.e. it can exist independently of any classes. On the other hand, in UML an association is defined in terms of association ends, which must be related to classifiers. Similar remarks apply to DAML (UML) DatatypeProperties versus UML attributes.

There are some advantages in using UML: the ontology languages are in general lacking in the expression of processes and behavior (UML provides sequence-diagrams, collaboration diagrams and activity diagrams). The are some disadvantages: UML is lacking of a formal semantic, is not *web-enabled* (e.g. is not based on XML, ...).

Among the proposals, we introduce two research projects building a tool to model ontologies by means of UML. ArgoUML [4] is a powerful yet easy-to-use interactive, graphical software design environment that supports the design, development and documentation of object-oriented software applications. ArgoUML (see figure 3) exports the created ontology into different languages and is an open source project.

Another project is the UML Based Ontology Tool-set (UBOT) [5] project building ontology engineering and natural language processing-based text annotation tools for DAML. UML is used as a front-end for visualizing and editing DAML ontologies. The approach is to extend UML by defining a prototype UML profile for DAML which maps UML stereotypes to DAML-specific elements [2]. The UBOT tools use Telelogic Tau UML Suite for editing and generating XMI that is translated to DAML. The UBOT project has been experimenting with formal methods to check the consistency of DAML ontologies. The UBOT tools are being evaluated in a satellite imagery analysis workflow agent application.

[4] http://argouml.tigris.org/

[5] http://ubot.lockheedmartin.com/

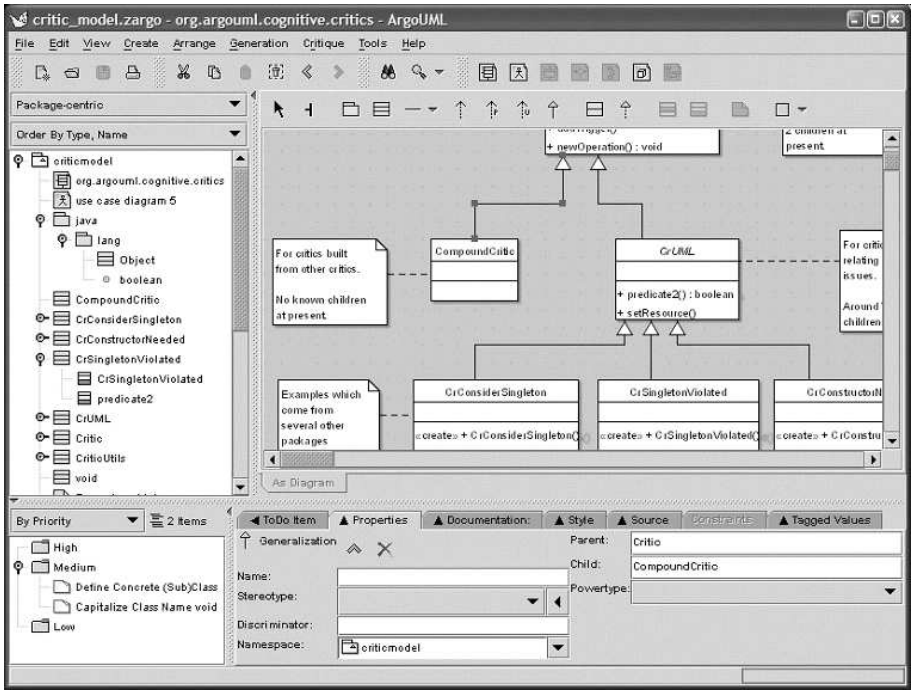

Figure 3: A screenshot of the ArgoUML tool

3.6 (Multi-) lingual issue

In order to be read by human reader, the ontology terms descriptions may be translated into different languages. There are two possible ways:

- Lexicalization of the terms: in the building phase, some descriptions for all the terms are provided in different languages;
- Each term of an ontology has to be mapped into an element of a lexical reference ontology (e.g. WordNet). The use of linguistic ontologies and multilingual ontologies is an integral component of an ontology management strategy, since it bridges the gap between linguistic terms and concepts in a domain. There are no common and proved definition for lexical ontology: in general with "lexical ontology" we mean "if the elements of an ontology (classes, properties, and individuals, possibly axioms) depend primarily on the acceptance of existing lexical entries, the ontology can be called "lexical". WordNet, formal or not, it's such a case"[6].

The DOGMA approach [16] is similar to the lexicalization approach: an ontology base is a set of context-specific binary fact types, called lexons: $<\gamma$: Term1,Role, Term2$>$. Here $\gamma \in \Gamma$ is just an abstract context identifier chosen from a set. The lexical terms (Term1, Role, Term2) are constructed from a given alphabet; for each $\gamma \in \Gamma$, and each term T occurring in a lexon, the pair (γ, T) specifies exactly a unique concept. The multi-lingual DOGMA approach [6, 5] consists in introducing a new linguistic identifier, called $l \in L$, where L is the linguistic space.

On the other hand, the MOMIS system [3] exploits WordNet (or Multi-WordNet[7]) in order to map for each class of the Global Virtual View a corresponding element belonging to WordNet. In this way a "well-known" meaning is assigned to each MOMIS concept. Moreover, by exploiting Multi-WordNet it is possible for each term to have a semi-automatic translation.

[6] Definition given by Gangemi in the "wordnet mailing list", 2005
[7] http://tcc.itc.it/projects/multiwordnet/multiwordnet.php

4 Ontology languages analysis

In this section, we propose a brief introduction to the main languages developed for defining ontologies. Such languages will be compared with each other in order to know the different capabilities and then classified on the basis of the criteria previously introduced.

4.1 Overview of the main ontology languages

Ontolingua. The term Ontolingua refers both to the system and to the language. The Ontolingua language is based on KIF (Knowledge Interchange Format) [12] and the Frame Ontology [14]. KIF has a declarative semantic and is based on 1rst-order predicate calculus. It provides definitions for object, function, relation and logical constants. KIF is a language for knowledge exchange, and is tedious to use for the development of ontologies. Thus, the Frame Ontology is built on top of KIF, and provides definitions for object-oriented and frame-language terms, like class, subclass-of,and instance-of.

OCML (Operational Conceptual Modeling Language). OCML was developed and is maintained by the Knowledge Media Institute (KMI) in context of the VITAL project [30]. Its primary purpose is to provide operational knowledge modeling facilities and to achieve this, it includes interpreters for functional and control terms. OCML provides mechanisms for defining relations, functions, classes, instances, rules and procedures. It can be viewed to some extent as "operational ontolingua", which provides theorem proving and function evaluation mechanisms for Ontolingua constructs. OCML provides a set of base ontologies that forms a rich modeling platform for building other ontologies: meta, functions, relations, sets, numbers, lists, strings, mapping, frames, inferences, environment and task-method.

LOOM. Loom [22] is a knowledge representation and reasoning system based on description logic. The University of Southern California's Information Sciences Institute (ISI) began the development in the late 80s, under DARPA sponsorship. A distinguished feature of description logic is that classes (concepts) can be defined in terms of descriptions that specify the properties or restrictions, which objects must satisfy in order to belong to the concept. One of the primary tasks of DL based system, like Loom, is to compute subsumption relationships between descriptions, and organize them into taxonomies. To achieve automatic derivation of taxonomies, Loom offers both a language for the description of objects and relationships, and an assertion language for specifying constraints on the concepts and relations. Loom provides powerful deductive reasoning with underlying production and classification-based inference capabilities.

F-logic (Frame Logic). F-logic [18] was developed in the late 80s. It is a logic language integrated with object-oriented or frame-based paradigm. Some fundamental concepts from object-oriented languages have a direct representation in F-logic, for example class, method, types and inheritance, and other secondary aspects, like polymorphism, can be easily modeled as well. There are many similarities between F-logic and Ontolingua, since they both try to integrate frames into logical framework. A difference is that F-logic lacks the powerful reification mechanism Ontolingua inherits from KIF, which allows the use of formulas as terms of meta-formulas.

XOL (XML-Based Ontology Exchange Language). XOL [17] was originally created to exchange ontologies for molecular biology. It provides a general definition that makes it appropriate for exchange of other ontologies as well. The modeling primitives and semantics are based on OKBC- Lite (a simplified form of OKBC knowledge model). In XOL, slots are to some extent treated as second-class citizens, which results in no support for slot hierarchies and weak

specification of relationships.

SHOE (Simple HTML Ontology Extensions). SHOE [21] is an extension of HTML to incorporate semantic knowledge in ordinary web documents by annotating html pages. SHOE provides modeling primitives to both specify and extend ontologies and to annotate web pages. Each page will declare which ontologies they are using, and thus make it possible for agents, which are aware of the semantics, to perform more intelligent searching. SHOE provides categories (classes), relations, inference rules, and rules to specify ontologies.

RDF (Resource Description Framework). RDF [20] is an infrastructure for encoding, exchange and reuse of structured metadata, proposed also by W3C. RDF provides a standard form for representing metadata in XML. The RDF data model consists of three object types: resources (subjects; available or imaginable entity), properties (predicates; describing the resources) and statements (objects; assigning a value for a property in a resource). RDF doesn't have any mechanisms for defining relationships between these, but the RDF Schema Specification Language (RDFS) does. RDFS can be used directly to describe ontologies, although its main intention is not for ontology specification. RDFS provides a set of modeling primitives for defining ontology (class, resource, property, is-a and element-of relationship etc.) and a standard way to encode them into XML. But RDFS has a rather limited expressive power, since axioms cannot be directly defined. Moreover, a number of other features are missing: it is not possible to declare local scopes of properties (range or domain restrictions that apply to some classes only), to define the disjointness of classes, to build new classes by combining other classes using union, intersection and complement, to define cardinality restrictions, and finally to describe special characteristic of properties (e.g. we may say that a property is transitive, unique or inverse of another property). We can see here the relation between ontology and RDF(s) is much closer than that between ontology and XML.

OWL (Ontology Web Language) [31] is a component of the Semantic Web activity. OWL makes an open world assumption. That is, descriptions of resources are not confined to a single file or scope. New information cannot retract previous information. New information can be contradictory, but facts and entailments can only be added, never deleted. The possibility of such contradictions is something the designer of an ontology needs to take into consideration. It is expected that tool support will help detect such cases. In order to write an ontology that can be interpreted unambiguously and used by software agents we require a syntax and formal semantics for OWL. OWL is a vocabulary extension of RDF.

ODL$_{I^3}$ [4]. ODL$_{I^3}$ is very close to the ODL language. ODL$_{I^3}$ is a source independent language used for information extraction to describe heterogeneous information in a common way. ODL$_{I^3}$ introduces the following main extensions with respect to ODL: Union constructor(to express alternative data structures), Optional constructor (to specify that an attribute is optional for an instance), Terminological relationships(they express intra and inter-schema knowledge). Rules (2 kinds of rules were introduced in ODL$_{I^3}$: if then and rules mapping rules)

4.2 Ontology languages analysis

In [8, 13], the authors provide a detailed comparison of the languages introduced in section 4.1. This comparison is carried out highlighting the capabilities of each language in satisfying the following criteria:

- Concepts (classes, objects or categories)
  - Partition definition: the possibility of an instance to be an instance of two concepts

belonging to a partition
  - Documentation definition: the possibility to include some comments to the ontology
  - Concept attribute definition (slots or functions or properties). Different kinds of attributes may be identified:
    * Instance attributes: whose value allow distinguishing a specific instance of a certain instance from other instances
    * Class attributes: whose value is attached to the concept
    * Local attributes: same name attributes attached to different concepts
    * Global attributes: whose domain is not specified
  - Predefined facets for attributes:
    * Default value slot
    * Type
    * Cardinality
    * Slot documentation

- Taxonomies (is-a, class inclusion, subsumption) on different kind of relations (generalization, specialization, subset hierarchy) according to the constraints involved in multiple taxonomic relationships (covering, partition, ...)
  - Subclass of (subsumption relationship)
  - Disjoint decomposition (partition where all concepts are subclasses of a common concept) (they can not be complete)
  - Exhaustive subclass decomposition: complete disjoint decomposition
  - Not subclass of

- Relations: interactions between concepts of the domain and attributes and Functions: special kind of relation where the value of the last argument is unique for a list of values of the n-1 preceding arguments
  - Arbitrary n-ary relation or function definition
  - Type of arguments constrained
  - Definition of integrity constraints in order to check the correctness of the arguments value
  - Operational definition to infer values of arguments with procedures, formulas, ...

- Axioms (or assertions): sentences that are always true

- Instances: elements in the domain attached to a specific concept
  - Instances of concepts definitions (membership conditions)
  - Instances of relations definitions (facts)
  - Claims (assertions of a fact made by an instance) definition

The results of this study is summerized in a table proposed in Figure 4. The table, extended from [8, 13] taking into account the ODL_{I^3} language, by means of an interesting synoptic view of the languages, points out that each language has a different expressive power, and consequently, it is suitable for a specific purpose.

Other researches propose different languages classifications. For example in [33] different analysis criteria are established: domain representation appropriateness, comprehensibility appropriateness and technical actor interpretation appropriateness. Figure 5 where S states for

structural, F functional , B behavioral, R rule, O object, C communication and AR actor-role, shows the results of the work.

The tables comparing languages demonstrate that the ontology languages focus os specific aspects and then the developer has to select the specific language for the specific goal. Next table offers a qualitative analysis for some of the proposed languages with respect to the six area we individuated in section. In particular we gave the same rate to ODL_{I^3} for the knowledge representation. RDF does not provide any mechanism to manage inference and reuse. ODL_{I^3} provides by means of the MOMIS tool an approach to the dynamics management but the language does not present any particular extension for it. On the other hand, OWL defines only a tag to manage the different versions. Finally, we gave a better rate to ODL_{I^3} in multi lingual management due to its intrinsic possibility to interact with (Multi-) WordNet.

	RDF	OWL	ODL_{I^3}
1) Domain Knowledge	+	+	+
2) Inference Mechanism	n.a.	+	+
3) Reuse and Integration	n.a.	-	-
4) Dynamic Management	n.a.	-	-
5) Graphical Modeling[8]	+	+	+
6) Multilingual management	n.a.	-	+

Table 1: Qualitative analysis on the basis of the six area individuated

Table 1 highlights that no language is able to cover all the requirements. For this reason, in order to chose the correct language, the developer has to analyze the domain where it will be used and the applications to be implemented, or hi has to extend a standard language.

5 Conclusions

The analysis of the developed languages and the papers comparing them indicate that no language fully satisfy the requirements for an ontology web language. Consequently, the evaluation for an ontology language is strongly connected to the use of the ontology.

The Wisdom goal is to develop intelligent techniques and tools, based on domain ontologies, to perform effective and efficient information search on the WEB. In particular, the Wisdom project aims at developing systems for retrieving information both from data-intensive and unstructured site/web pages, in an integrated and efficient way.

For these reasons the ontology language has to be expressive enough to represent mappings between heterogeneous independently developed ontologies and has to manage the ontology evolution due to the integration of a new information source. Deliverable D2.R1, Critical analysis of languages and mapping techniques, analyzes the requirements for a mapping language, D1.R2: Definition of the language and techniques for domain ontology specification will define the language for the project.

Finally, independently of the language will be chosen for Wisdom, the interoperability with other languages and sources is an importantant task. With reference with ODL_{I^3} the interoperability with the W3C standard languages is guaranteed by means of translation rules, we propose in appendix A.

References

[1] G. Antoniou and F. van Harmelen. *A semantic web primer*. The MIT Press, ISBN 0-262-01210-3, Cambridge Massachusetts, 2004.

[2] K. Baclawski, M. M. Kokar, P. A. Kogut, L. Hart, J. E. Smith, J. Letkowski, and P. Emery. Extending the unified modeling language for ontology development. *Software and System Modeling*, 1(2):142–156, 2002.

[3] D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. Synthesizing an integrated ontology. *IEEE Internet Computing Magazine*, pages 42–51, September-October 2003.

[4] S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic integration of heterogeneous information sources. *Data & Knowledge Engineering, Special Issue on Intelligent Information Integration*, 36(1):215–249, 2001.

[5] J. De Bo and P. Spyns. Extending the dogma framework in view of multilingual ontology integration. In *Technical Report 09, STAR Lab, Brussel*, http://www.starlab.vub.ac.be, 2003.

[6] J. De Bo, P. Spyns, and Robert Meersman. Creating a "dogmatic" multilingual ontology infrastructure to support a semantic portal. In *Proceedings of OTM Confederated International Workshops, Catania, Sicily, Italy*, pages 253–266, 2003.

[7] P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. Contextualizing ontologies. *Journal of Web Semantics*, 1(4):325–343, 2004.

[8] Ó. Corcho, M. Fernàndez-López, and A. Gómez-Pérez. Methodologies, tools and languages for building ontologies: Where is their meeting point? *Data & Knowledge Engineering*, 46(1):41–64, 2003.

[9] Ó. Corcho and A. Gómez-Pérez. A roadmap to ontology specification languages. In *Proc. of 12 International Conference on Knowledge Engineering and Knowledge Management EKAW*, pages 80–96, Juan-les-Pins,French Riviera, 2000.

[10] M. Dorr, N. Guarino, M. Fernandez-Lopez, E. Schulten, M. Stefanova, and A. Tate. State of the art in content standards. In *Deliverable 3.1, OntoWeb project*, http://www.ontoweb.org, 2002.

[11] A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari. Sweetening wordnet with dolce. *AI Magazine*, 24(3):13–24, 2003.

[12] M. R. Genesereth and R.E. Fikes. Knowledge interchange format version 3.0 reference manual.

[13] A. Gómez-Pérez and Ó. Corcho. Ontology specification languages for the semantic web. *IEEE Intelligent Systems*, 17(1):54–60, 2002.

[14] T.R. Gruber. A translation approach to portable ontology specification,. *Knowledge Acquisition*, 5:199–220, 1993.

[15] N. Guarino. Formal ontology in information systems. In *Proceedings of Formal Ontology in Information Systems 1998*, pages 3–15, Trento, Italy, 1998.

[16] M. Jarrar and R. Meersman. Formal ontology engineering in the dogma approach. In *DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*, pages 1238–1254, Irvine, California, USA, 2002. Springer.

[17] R. Karp, V. Chaudhri, and J. Thomer. XOL: An xml-based ontology exchange language, 1999. Available at http://www.ai.sri.com/ pkarp/xol.

[18] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. In *Journal of the ACM*, 1995.

[19] M. Klein and D. Fensel. Ontology versioning on the semantic web. In *In International Semantic Web Working Symposium (SWWS)*, Stanford University, USA, August 2001.

[20] O. Lassila and R. R. Swick. Resource description framework (RDF) model and syntax specication, 1999. W3C Recommendation 22 February 19. Available at http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[21] S. Luke and J. Heflin. SHOE 1.01 proposed specification, 2000. Available at http://www.cs.umd.edu/projects/plus/SHOE/spec.html.

[22] R. MacGregor. Using a description classier to enhance deductive inference. In *Proceedings of Seventh IEEE Conference on AI Application, Florida*, pages 93–97, 1995.

[23] B. Motik, N. Stojanovic, L. Stojanovic, and A. Maedche. User-driven ontology evolution management. In *In 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW2002)*, Siguenza, Spain, October 2002.

[24] I. Niles and A. Pease. Towards a standard upper ontology. In *Proceedings of 2nd International Conference on Formal Ontology in Information Systems, FOIS 2001, Ogunquit, Maine, USA, October 17-19*, pages 2–9, 2001.

[25] N. F. Noy. A survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, 2004.

[26] N. F. Noy and M. C. A. Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 6(4):428–440, 2004.

[27] N. F. Noy and D. L. McGuinness. Ontology development 101: A guide to creating your first ontology. In *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*, http://www.ksl.stanford.edu/people/dlm/publications.html, 2001.

[28] OMG, editor. *Ontology Definition Metamodel Request For Proposal OMG Document.* OMG Document: ad/2003-03-40, available at http://www.omg.org, 2003.

[29] R.A. Pottinger and P. A. Bernstein. Merging models based on given correspondences. In *Proceedings of 29th International Conference on Very Large Data Bases, September 9-12, 2003, Berlin, Germany*, pages 826–873, 2003.

[30] N. Shadbolt, E. Motta, and A. Rouge. Constructing knowledge based systems. *IEEE Software*, 10(6):34–38, 1993.

[31] M. Smith, C. Welty, and D. L. McGuinness. OWL Web Ontology Language Guide, 2004. W3C Recommendation 10 February 2004. Available at http://www.w3.org/TR/owl-guide/.

[32] R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25(1-2):161–197, 1998.

[33] X. Su and J. A. Gulla. Semantic enrichment for ontology mapping. In *In Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems*, Manchester, UK, 2005.

[34] X. Su and L. Ilebrekke. A comparative study of ontology languages and tools. In *Proc. of 14th Conference on Advanced Information Systems Engineering - Doctoral Consortium (CAiSE '2002)*, pages 761–765, Toronto, Canada, 2002.

[35] V. Tamma. *Theoretical foundations of ontologies. An Ontology Model supporting Multiple Ontologies for Knowledge sharing.* PhD Thesis, University of Liverpool, 2001.

[36] M. Uschold and M. Grninger. Ontologies and semantics for seamless connectivity. *SIGMOD Record*, 33(4):58–64, 2004.

[37] C. A. Welty and N. Guarino. Supporting ontological analysis of taxonomic relationships. *Data & Knowledge Engineering*, 39(1):51–74, 2001.

[38] Y.Kalfoglou and W. M.Schorlemmer. Ontology mapping: The state of the art. In *Semantic Interoperability and Integration*, volume 04391 of *Dagstuhl Seminar Proceedings*. IBFI, Schloss Dagstuhl, Germany, 2005.

A Comparison between ODL$_{I^3}$ and OWL

The following tables show a comparison between the ODL$_{I^3}$ language and OWL both Lite and DL. In particular, we show the main difference (within OWL Lite many operators similar to ODL$_{I^3}$ operators are not defined) and we define the rules to translate one language into another one. By means of the prefix *sew:* we define the new features needed to translate all the terms.

Terms ODL$_{I^3}$	**Terms OWL DL**	**Terms OWL Lite**	**Logic Interpretation**
–	owl:Thing	owl:Thing	set of all the instances
–	owl:Nothing	owl:Nothing	empty set
interface	owl:Class	owl:Class	class concepts (intensional)
view	owl:Class	owl:Class	view concept
isa	rdfs:subClassOf rdfs:subPropertyOf	rdfs:subClassOf (classes or restrinctions)	estensional hierarchy
nt$_{ext}$ bt$_{ext}$	rdfs:subClassOf rdfs:subPropertyOf	rdfs:subClassOf (classes or restrinctions)	estenstiona hierarchy
syn$_{ext}$	owl:equivalentClass owl:equivalentProperty	owl:equivalentClass (classes or restrinctions)	equivalence estensional
and	owl:intersectionOf (classes or restrinctions)	owl:intersectionOf	intersection
union	owl:unionOf	–	union
enum	owl:DataRange...owl:oneOf ... rdf:List...rdf:Rest	–	enumeration
range	owl:DataRange...owl:oneOf ... rdf:List...rdf:Rest (restricted range)	–	range
–	owl:complementOf	–	negation
–	owl:disjointWith	–	disjunction estentional
bt	sew:ThesRelation... sew:RelType...bt	sew:ThesRelation... sew:RelType...bt	intensional Hypernym (ODL$_{I^3}$)
nt	sew:ThesRelation... sew:RelType...nt	sew:ThesRelation... sew:RelType...nt	intensional hyponym (ODL$_{I^3}$)
rt	sew:ThesRelation... sew:RelType...rt	sew:ThesRelation... sew:RelType...rt	association (ODL$_{I^3}$)
syn	sew:ThesRelation... sew:RelType...syn	sew:ThesRelation... sew:RelType...syn	intensional synonym (ODL$_{I^3}$)

Table 2: Comparison between ODL$_{I^3}$ and OWL

Terms ODL$_{I^3}$	**Terms OWL DL**	**Terms OWL Lite**	**Logic Interpretation**
–	owl:sameAs (for instances only)	owl:sameAs (for instances only)	equivalence among instances
–	owl:differentFrom	owl:differentFrom	difference from instances
–	owl:AllDifferent... owl:distinctMembers	owl:AllDifferent... owl:distinctMembers	difference from instances
attribute DomainType "Interface or View"	owl:ObjectProperty	owl:ObjectProperty	relationship between instances
attribute DomainType "DataType"	owl:DatatypeProperty	owl:DatatypeProperty	relationship between instances and a value
attribute	rdfs:domain	rdfs:domain (classes only)	property subject
attribute Domain	rdfs:range	rdfs:range (only classes)	property object
–	rdfs:subPropertyOf	rdfs:subPropertyOf	property hierarchy
–	owl:equivalentProperty	owl:equivalentProperty	estensional equivalence among properties
relationship...inverse	owl:inverseOf (Object properties)	owl:inverseOf (Object properties)	inverse property
relationship...inverse	owl:SymmetricProperty (Object properties)	owl:SymmetricProperty (Object properties)	symmetric property (equal to the inverse property)
key (single)	owl:FunctionalProperty + owl:Cardinality (1)	owl:FunctionalProperty + owl:Cardinality (1)	functional property cardinality restriction
key (multiple values)	sew:Key	sew:Key	concetto di chiave (ODL$_{I^3}$)
foreign key... references	rdfs:subPropertyOf KEY Properties	rdfs:subPropertyOf KEY Properties	key concept foreign key (ODL$_{I^3}$)
–	owl:TransitiveProperty (Object properties)	owl:TransitiveProperty (Object properties)	transitive property
*	owl:minCardinality = 0 owl:maxCardinality = 1	owl:minCardinality = 0 owl:maxCardinality = 1	optional attribute

Table 3: Comparison between ODL$_{I^3}$ and OWL

Terms ODL$_{I^3}$	Terms OWL DL	Terms OWL Lite	Logic Interpretation
attribute set forall	owl:allValuesFrom (classes or data range)	owl:allValuesFrom (classes)	universal quantification
exists	owl:someValuesFrom (classes or data range)	owl:someValuesFrom (classes)	esistential quantificator
–	owl:hasValue	–	universal instance quantification
–	owl:minCardinality	owl:minCardinality(0,1)	minimal cardinality
–	owl:maxCardinality	owl:maxCardinality(0,1)	maximal cardinality
FixedArraySize	owl:Cardinality	owl:Cardinality(0,1)	maximal and minimal cardinality
rule	owl:Restriction... owl:onProperty	owl:Restriction... owl:onProperty	properties restriction
mapping rule	URI reference	URI reference	relationship between integrated schema and local sources
–	owl:AnnotationProperty	owl:AnnotationProperty	
–	owl:versionInfo	owl:versionInfo	Properties for the document
–	rdfs:label	rdfs:label	annotation with respect to
–	rdfs:comment	rdfs:comment	the Dublin Core metadataset
–	rdfs:seeAlso	rdfs:seeAlso	
–	rdfs:isDefinedBy	rdfs:isDefinedBy	
–	owl:Ontology... owl:imports	owl:Ontology... owl:imports	importing of an ontology (transitive)
–	owl:Ontology... owl:priorVersion	owl:Ontology... owl:priorVersion	reference to the previous version of an ontology
–	owl:Ontology... owl:backwardCompatibleWith	owl:Ontology... owl:backwardCompatibleWith	reference to a consistent version of an ontology
–	owl:Ontology... incompatibleWith	owl:Ontology... incompatibleWith	reference to a not consistent ontology

Table 4: Comparison between ODL$_{I^3}$ and OWL

Terms ODL$_{I^3}$	Terms OWL DL	Terms OWL Lite	Logic Interpretation
–	owl:DeprecatedClass	owl:DeprecatedClass	deprecated class
–	owl:DeprecatedProperty	owl:DeprecatedProperty	deprecated property
	sew:lemmaValue	–	
	sew:lemmaSyntacticCategory	–	terminological
wnAnnotation	sew:lemmaSenseNumber	–	annotation
	sew:nodeUri	–	respect WordNet
	sew:ontoVersionInfo	–	
	sew:ontoCreator	–	

Table 5: Comparison between ODL$_{I^3}$ and OWL

	Onto-lingua	OCML	LOOM	FLogic	XOL	SHOE	OML	RDF(S)	OWL	ODLI3
Concepts General Issues										
Partitions	+	+-	+	+-	-	-	+	-	+	+
Documentation	+	+	+	+-	+	+	+	+	+	+
Attributes										
Instance attribute	+	+	+	+	+	+	+	+	+	+
Class attribute	+	+	+	+	+	-	+	-	+	+
Local scope	+	+	+	+	+	+	+	+	+	+
Global scope	+-	+-	+	-	+	-	+	+	+	+
Facets										
Default slot value	-	+	+	+	+	-	-	-	-	-
Type	+	+	+	+	+	+	+	+	+	+
Cardinality	+	+	+	+-	-	-	+	-	+	-
Slot doc.	+-	+-	+	-	+	+	+	+	+	+
Taxonomies										
Subclass of	+	+	+	+	+	+	+	+	+	+
Exhaustive decomposition	+	+-	+	+-	-	-	+-	-	+	-
Disjoint decomposition	+	+-	+	+-	-	-	+	-	+	-
Not subclass of	+-	-	+-	-	-	-	-	-	+	+-
Relations and Functions										
n-ary relations/ functions	+	+	+	+-	+	+	+	+-	+-	+-
Type constraints	+	+	+	+	+	+	+	+	+	+
Integrity constraints	+	+	+	+	-	-	-	-	-	-
Operational definitions	-	+	+	+	-	-	-	-	-	+
Axioms										
1st order logic	+	+	+	+-	-	-	-	-	+	+
2nd order logic	-	-	-	-	-	-	-	-	-	-
Independent axioms	+	+	-	-	-	-	-	-	-	+
Embedded axioms	+-	+	+	-	-	-	+	-	-	+
Instances										
Instances of concepts	+	+	+	+	+	+	+	+	+	+-
Facts	+	+	+	+	+	+	+	+	+	+-
Claims	-	-	-	-	-	+	-	+-	+-	-

Figure 4: Languages comparison extended from [13]

		CycL	Ontolingua	F-Logic	OCML	LOOM	Telos	RDF(S)	OIL	DAML+OIL	XOL	SHOE
Domain appropriateness	Expressive Power	High	High	Medium	Medium+	Medium	High	Medium-	Medium-	Medium+	Medium	Medium
	Perspectives	S,O-,R	S,O+,R	S,O+,R	S-,O,R, F,	S,O+,R,	S,O,R+ F,AR-	S,O,R-	S,O,R-	S,O,R-	S,O,R-	S,O,R
Comprehensibility appropriateness	Number of Constructs	Large	Large	Medium	Medium+	Medium	Medium+	Small	Small	Medium-	Medium	Small
	Abstraction Mechanism	Cla Gen+ Agg Ass	Cla Gen+ Agg Ass	Cla Gen+ Agg Ass	Cla Gen+ Agg- Ass	Cla Gen+ Agg- Ass	Cla Gen+ Agg Ass	Cla Gen- Agg- Ass	Cla Gen Agg- Ass	Cla Gen Agg- Ass	Cla Gen- Agg- Ass	Cla Gen- Agg- Ass
Technical actor interpretation appropriateness	Formal Syntax	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Formal Semantics	Yes	Yes	Yes	Yes	Yes	Yes	Yes-	Yes	Yes-	No	Yes-
	Inference Engine	Weak	No	Good+	Good	Good+	Good	No	Good+	Possible	No	Good
	Constraint Checking	Good	Good	Good	Good	Good	Good	Weak	Weak	Weak	Weak-	Weak

Figure 5: Languages comparison in [33]