Ontology evaluation and validation

An integrated formal model for the quality diagnostic task

Aldo Gangemi[1], Carola Catenacci[1], Massimiliano Ciaramita[1], Jos Lehmann[1]

contributions by:
Rosa Gil[2] (*in section 2.2*)
Francesco Bolici[3] and Onofrio Strignano[3] (*in section 2.4*)

[1]Laboratory for Applied Ontology, ISTC-CNR, Roma/Trento (Italy)
{aldo.gangemi, carola.catenacci, jos.lehmann, m.ciaramita}@istc.cnr.it
[2]Technology Department, Universitat Pompeu Fabra, Barcelona (Spain)
rosa.gil@upf.edu
[3]Research Centre on Information Systems, Università 'Guido Carli' LUISS (Italy)
{fbolici, ostrignano}@luiss.it

Introduction

The need for evaluation methodologies in the field of ontology development and reuse showed up as soon as 1994 and it has been growing ever since [Sure 2004]. Yet, no comprehensive and global approach to this problem has been proposed to date.
What we present here is a novel perspective that tries to integrate the varied (and differently aimed) ontology evaluation methods proposed so far. Particular efforts are devoted to considering as many as possible measures that can affect the quality of an ontology, and whose metrics can either be qualitative or quantitative, with the ultimate goal to design a formal model for ontology evaluation.
The important topic of evaluation of *tools* for ontology realization is not addressed in the present contribution, and we plan to consider it in future work.
Our contribution is structured as follows. In the first section, we introduce a metaontology (provisionally called here O^2) which characterizes ontologies as semiotic objects and that is meant to provide a meta-theoretical foundation to ontology evaluation and annotation.
In the second section, such a metaontology is complemented with an ontology of ontology evaluation and validation (called *oqval*), which picks up ontology elements based on the metaontology, and provides quality parameters (and possible ordering functions) for them.
In practice, we model ontology evaluation as a *diagnostic task* involving ontology descriptions, which, in turn, include the roles and functions of the elements in an ontology, the parameters assumed within those descriptions that denote the "quality" of an ontology, and some functions that compose those parameters according to a preferential ordering.
We identify and discuss three distinctions over measure types for ontology evaluation which are based on O^2, i.e. *structural*, *functional* and *usability-related* measures, and provide some examples of preferential order over measures.
Finally, in the third section, we provide a comparing review of a selection of current literature on ontology evaluation.

1. A semiotic metaontology

We consider an ontology a semiotic object, i.e. an object constituted by an *information object* and an *intended conceptualization* established within a *communication setting*.
The basic intuition is that information is any pattern that *can* represent another pattern, whereas that representation is interpretable by some rational agent (this intuition comes back at least to C.S. Peirce [Peirce 1931]).
An ontology is a special kind of information, whose patterns are graph-like structures, and whose represented patterns are intended conceptualizations, i.e. internal representations of (mainly) *types of things*. For example, one can typically define an ontology for subways, but one will hardly consider the London Underground graph an ontology (it would be eventually considered a *model* of an appropriate subway ontology).
In semiotics, e.g. [Eco 1984], the information object (e.g. the OWL(DL) version of FOAF -Friend Of A Friend- ontology[1]) is said to play the role of *expression*, the intended conceptualization (e.g. the conceptual relation between persons, their addresses, and the knowledge of each other) is said to play the role of *meaning*, and the communication setting is said to play the role of *context* (e.g. the task, application, and usage context of FOAF). These roles are complemented by other communication elements defined by [Jakobson 1960] (Fig.1): one or more *rational agent(s)* (e.g. a FOAFer) who play(s) the role of *interpreter*, a *medium* (e.g. the Web) that plays the role of *channel*, and an *encoding system* (e.g. OWL) that plays the role of *code*. Moreover, when a medium is used to realize an information object through a channel, an *information realization* appears (e.g. a graphic visualization of the FOAF ontology on a screen).

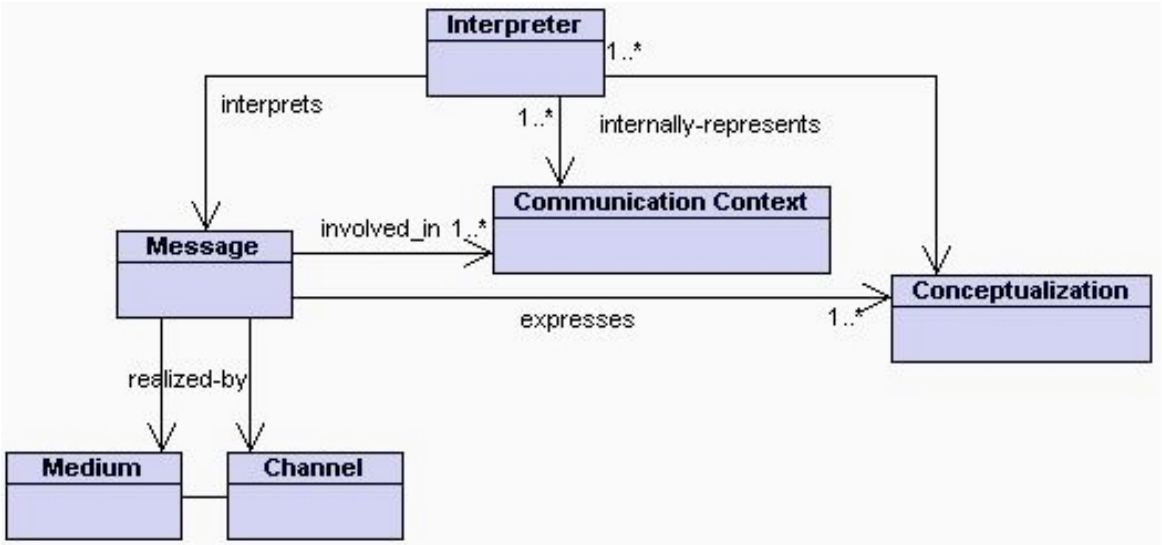

Figure 1. The communications roles in ontology engineering, adapted from [Jakobson 1960]. Messages are information objects, communication contexts and conceptualizations are (cognitive) descriptions. Media and channels realize the messages.

An ontology as a semiotic object has therefore several interdependent features that can be measured, with several possible quality parameters. Following the example, we could measure the abstract information contained in the FOAF ontology graph, the capability of that information pattern to mirror a "social relationships" pattern, the usability of FOAF ontology for a certain application, its accessibility by agents looking for abstract information of that kind, its provenance, etc.

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

We provide a minimal FOL formalization of an ontology as a semiotic object. For that purpose, we reuse our ontology of information objects (denoted by the prefix: "inf") [Gangemi et al. 2004], built upon DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering, denoted by the prefix "dol") [Masolo et al 2004] and ExtendedDnS (a theory of descriptions and situations, denoted by the prefix "edns") [Gangemi et al. 2004].
Such *meta-ontology*, provisionally called here O^2 (Fig.2), is complemented with an ontology of ontology evaluation (called *oqval*, Fig.3), which represents ontology elements based on the metaontology, and also allows the representation of quality parameters (and possible ordering functions) for them.

Structurally, an ontology is any graph whose nodes and arcs represent conceptualizations, independently from how these conceptualizations can be given a formal semantics. In (even) more informal terms, an ontology is a graph of metadata (i.e., from thesauri to taxonomies and axiomatized theories).
We further stipulate that ontology graphs are directed along the arcs representing a *subClassOf* (or *isa*) relationship.
Breadth- and depth-oriented measures on the graph are made on *isa* (taxonomical) relationships as well.
On the contrary, density-oriented measures are made on *non-isa* relationships.
We use acronyms as prefixes (see above for acronym explanation) to denote the ontology of reused predicates.

In O^2, an ontology graph *OG* is an edns:Information-Object, its possible formal semantics (the ontology semantics *OS*) is a dol:Abstract entity (a space), and its intended meaning (the ontology intended conceptualization *OC*) is a edns:Theory (intended as a kind of edns:description).
Given an ontology, a formal semantic framework (e.g. model-theoretic semantics), a context of ontology production, and at least one context of ontology use, we can propose the following axioms to give a (first-order, reified) meta-theoretical foundation to ontology evaluation and annotation:

(A1) $OG(x) \rightarrow$ edns:InformationObject(x)
(A2) $OS(x) \rightarrow$ dol:Abstract(x)
(A3) $OC(x) \rightarrow$ edns:Description(x)

A1-A3 are based on the DOLCE and ExtendedDnS ontologies:
- edns:InformationObject is a sub-class of non-agentive social objects and includes any symbolic entity, independently from its concrete realization
- edns:Description is a sub-class of non-agentive social objects and includes the reifications of relations of any kind. By assuming that an intended conceptualization can be reduced to a conceptual relation, descriptions are used as ontology placeholders for intended conceptualizations
- dol:Abstract is a primitive from DOLCE, and includes all entities that do not have a spatio-temporal localization (i.e. they are regions in an abstract space, e.g. values, sets, etc.).

Ontologies as information objects vs. abstract spaces :

(A4) $OG(x) \rightarrow \exists y(\text{inf:q-represents}(x,y) \wedge OS(y))$

A4 reuses the relation *q-represents*(*x,y*), defined in the information objects ontology, which encodes the fact that information objects can reify regions in abstract spaces. A4 means that an ontology graph is supposed to have a mapping in an abstract space (a formal semantics in the typical case, e.g. if using OWL).
Notice that the main difference between an abstract space and an information object is that the second has a spatio-temporal localization. E.g. given OWL, all ontologies that can be built according to the OWL datamodel always exist in an OWL abstract space, but a given OWL ontology has a social existence due to its time of creation, its creators, its usage, etc.
On the other hand, A4 is too strong for the ontologies that are not defined in a logical language with an explicit formal semantics. Hence, we make the further assumption that any ontology can be *reengineered* in order to get (partly or completely) a formal semantic characterization.

Ontology graphs express meaning :

(A5) $OG(x) \rightarrow \exists y(\text{edns:expresses}(x,y) \wedge OC(y))$

A5 reuses the relation *expresses*(*x,y*), defined in the ExtendedDnS ontology, which encodes the fact that information objects express intended meanings (in this context: descriptions). A5 means that an ontology graph necessarily expresses at least one intended meaning.
Notice that in the real world, an information object (hence an ontology) can be used to provide additional expression to an existing conceptualization, for example, when *profiling* an ontology, or when formalizing a natural language explanation.

Semantic adequacy: ontology formal semantics should catch intended meaning:

(A6) $OS(x) \rightarrow (\exists y,z(OC(y) \wedge OG(z) \wedge \text{edns:expresses}(z,y) \wedge \text{inf:q-represents}(z,x)) \rightarrow \text{edns:admitted-by}(x,y))$

A6 reuses *admitted-by*(*x,y*), a composed relation from the ExtendedDnS ontology, which encodes the fact that a description can define a *logical parameter* that can be valued by some value from a certain value set (i.e. a formal semantic space). A6 means that the formal semantics of an ontology is admitted by its intended meaning if the formal space is q-represented by an ontology graph that also expresses that intended meaning.
In other words, A6 asserts that semantic adequacy of an ontology requires compliance to intended meanings, not just formal validity.

(T1) $OC(x) \rightarrow (\exists y,z(OS(y) \wedge OG(z) \wedge \text{edns:expresses}(z,x) \wedge \text{inf:q-represents}(z,y)) \rightarrow \text{edns:admitted-by}(y,x))$

From A6, and the axioms of ExtendedDnS and information objects ontology, we can infer that the intended meaning of an ontology graph that q-represents a formal space, also admits that formal space as a value for its logical parameter.

(T2) $OG(x) \rightarrow (\exists y,z(OS(y) \wedge OC(z) \wedge \text{edns:admitted-by}(y,z) \wedge \text{edns:expresses}(x,z)) \rightarrow \text{inf:q-represents}(x,y))$

From A6, and the axioms of ExtendedDnS and information objects ontologies, we can infer that an ontology graph that expresses an intended meaning that admits a value from a formal semantic space, also q-represents that space.

(D1) o-interprets(x,y) $=_{df}$ inf:interprets(x,y) $\wedge$ edns:RationalAgent(x) $\wedge$ OG(y) $\wedge$ $\exists z$(edns:internally-represents(x,z) $\wedge$ OC(z) $\wedge$ edns:expresses(y,z))

D1 defines a relation holding between rational agents and ontology graphs, and requires that the rational agent internally represents the ontology intended meaning expressed by the graph.

(D2) o-encodes(x,y) $=_{df}$ o-interprets(x,y) $\wedge$ $\exists z$(edns:creates(x,z) $\wedge$ OC(z) $\wedge$ edns:expresses(y,z))

D2 defines a sub-relation of o-interprets, in which the rational agent must be the creator of the intended meaning.

(A7) OG(x) $\rightarrow$ $\exists y$(o-encodes(y,x))

A7 states that an ontology graph has at least one encoder.

(T3) OG(x) $\rightarrow$ $\exists y$(o-interprets(y,x))

T3 holds after A7 and D2: an ontology graph has at least one interpreter.

(D3) o-decodes(x,y) $=_{df}$ o-interprets(x,y) $\wedge$ $\exists z$(o-encodes(z,y))

D3 defines a sub-relation of o-interprets, in which there must be a (not necessarily different) rational agent that is the creator of the intended meaning.

(D4) profiles(x,y) $=_{df}$ edns:InformationObject(x) $\wedge$ OG(y) $\wedge$ $\exists z,w,a$(OC(z) $\wedge$ edns:expresses(x,z) $\wedge$ edns:expresses(y,z) $\wedge$ edns:Description(w) $\wedge$ edns:expresses(x,w) $\wedge$ $\neg$(edns:expresses(y,w)) $\wedge$ o-decodes(a,y) $\wedge$ o-encodes(a,x))

D4 defines a relation holding between information objects, in which the second one is an ontology graph, while the first (the "ontology profile") also expresses the same intended meaning as the second does, but it also expresses meta-level knowledge about the communication setting, and the profile is o-encoded by a rational agent that also o-decodes the ontology graph.
The UML class diagram in Fig.2 summarizes the axioms given above as a design pattern: an ontology graph has an intended conceptualization and a formal semantic space admitted by the logical parameter of the intended meaning. The graph and the meaning are kept together by a rational agent. An agent can also provide a profile of the structural and functional properties of an ontology graph in order to enhance or to enforce its usability.

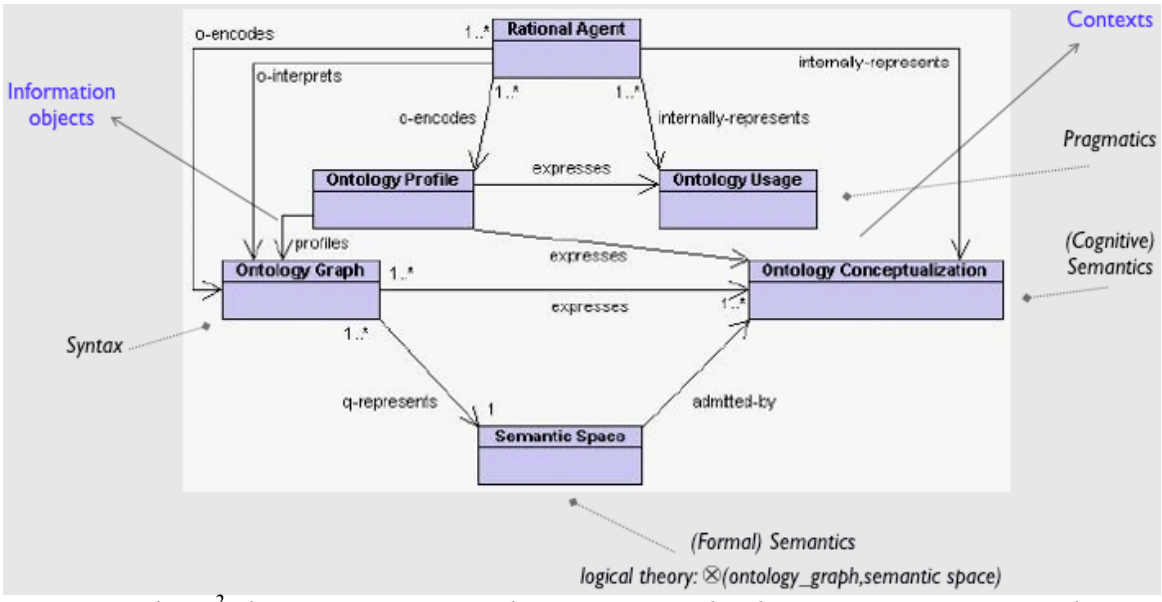

Figure 2. The O^2 design pattern. Ontologies are graphs that express a conceptualization and can be profiled by additional information that expresses their usage context. An ontology graph has ("q-represents") a formal semantics if it ithat can bes admitted by the conceptualization. These constraints are the sensible part of ontology evaluation: does the formal semantics catch the intended conceptualization (the "cognitive" semantics)?

Based on O^2, we identify three distinctions over measure types for ontology evaluation, i.e.:

- *structural measures*, which focus on the syntax of ontology graphs (mainly, but not exclusively, without referring to its intended meaning, semantics, and context) and their possible formal (abstract) semantics (that can be considered an additional syntax)
- *functional measures*, focusing on the relations holding between the ontology graph and its intended meaning (i.e. on the cognitive semantics). All these are precision/recall-based measures, but there are many specific issues concerning what data should be matched to obtain the measures
- *usability-profiling measures*, focusing on the ontology profile, which typically addresses the communication context of an ontology (i.e. its pragmatics). These measures focus on ontology annotations, i.e. metadata about *recognition*, *economical efficiency*, and *interfacing* of an ontology. Annotations typically contain information about structural or functional properties of an ontology.

2. A model of ontology evaluation and validation *(oqual)*

We model ontology evaluation and validation (together) as a *diagnostic task* (Fig.s 3,4) involving:

- *quality-oriented ontology descriptions (goods)*, which provide the *roles* and *functions* of the elements from an ontology, and have elementary qoods (called "principles") as parts
- *value spaces* ("attributes") of ontology elements, bearing typical dependencies on other spaces
- *principles* for assessing the ontology fitness, which are modelled as elementary quality-oriented ontology descriptions, and are typically parts of a project-oriented qood
- *parameters* (ranging over the attributes -*value spaces*- of ontologies or ontology elements), defined within a principle
- *parameter dependencies* occurring across principles because of the interdependencies between the value spaces of the measured ontology elements
- *preferential ordering* functions that compose parameters from different principles
- *trade-offs*, which provide a conflict resolution description when combining principles with conflicting parameters (see section 2.5).

The formal model of the ontology diagnostic task, called *oqual*, is based on the same ontology design pattern used for O^2 (the *Description*<->*Situation* pattern from the ExtendedDnS ontology [Gangemi 2005]).[2]

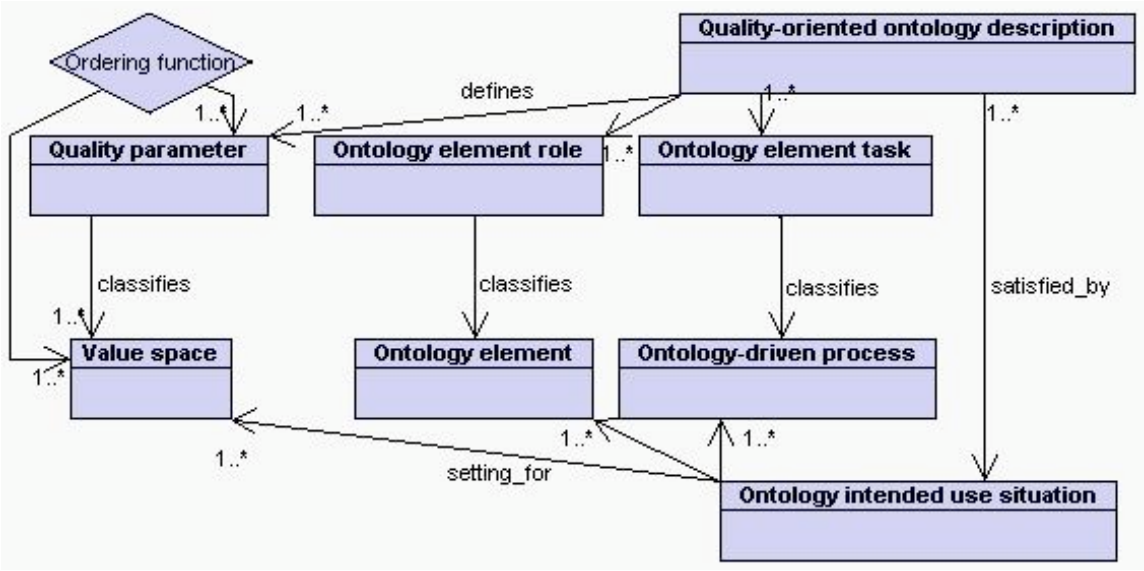

Figure 3. The ontology evaluation design pattern (oqual)

Ontology descriptions, roles, parameters, and preferential orders are obtained by looking at the measure types that can be performed on an ontology, which can be characterized according to several methods.
An example of how *oqual* can be applied is presented in Fig.4.

[2] An axiomatization similar to that given for O^2 will be provided in the next version.

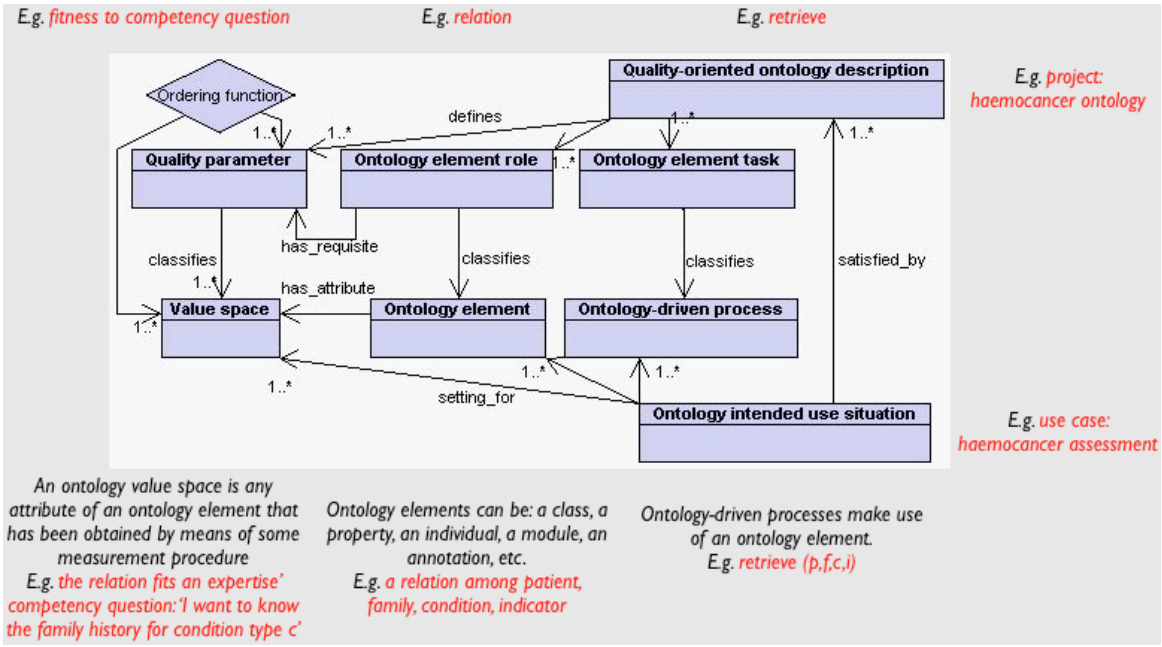

Figure 4. Applying oqval to a clinical use case.

Before discussing the measure types, we clarify our model of the diagnostic task.

2.1 Measure types

Our analysis of ontology evaluation addresses the problem in terms of answers to three main questions:

a) What to measure in an ontology? and how? the quality of an ontology may be assessed relatively to various dimensions. As explained above (see Section 1), by ontology we mean any kind of graph of metadata, and we propose to measure its quality relatively to three main groups of dimensions: structural, functional and usability-related dimensions.

1) An ontology shows its *structural* dimensions when represented as a graph. In this form, the topological and logical properties of an ontology may be measured by means of a metric. The existence of these structural dimensions, however, can be considered independent from the metric is being used.
2) The *functional* dimensions are related to the intended use of a given ontology and of its components, i.e. their function. Functional dimensions are things like agreement, task, topic, design, etc. Such dimensions become apparent in an ontology depending on the context, which in turn is given by the way in which the ontology is chosen, built, exploited, etc. In our intuition, functional dimensions are (relational, hence) extrinsic to the ontology graph.
3) Finally, *usability-related* dimensions depend on the level of annotation of a given ontology. How easy it is for users to recognize its properties? How easy is to find out which one is more (economically, computationally) suitable for a given (series of) task(s)?

Notice that these dimensions follow a partition into *logical types*: structurally, we look at an ontology as an (information) object; functionally, we look at it as a language

(information object+intended conceptualization), and from the usability viewpoint, we look at its meta-language (the profile about the semiotic context of an ontology). Therefore, the dimension types correspond to the constituents of an ontology as a semiotic object (with the notable issue of a possible mismatch between formal and cognitive semantics).
Heterogeneous methods are needed, and some measures can result from more than one method. See 2.2-2.4 for a list of measurement methods.

b) Which parameters for the quality of an ontology? Each measure can have more than one quality parameter, depending on other parameters/measures, and the overall composition for a given ontology project implies a non-linear procedure to quality assessment.
For example, in an ontology project we may want to combine measures like *logical complexity* and *presence of dense areas* (e.g. of *design patterns*). If *high density* is chosen as a quality parameter, then the parameter associated with *high complexity* is chosen too, because usually dense areas involve a lot of restrictions, sometimes with indirect cycles; in other words, high-density parameter depends on the high-complexity parameter (see section 2.5). On the other hand, if the quality parameter is *lower complexity*, then the parameter associated with *lower density* is chosen too, because the first depends on the second.
Actually, this is an application of a general pattern of parameter composition ranging on mutually dependent scalar spaces: when we compose two parameters p_1 and p_2 ranging respectively on value spaces s_1 and s_2 with a scalar metric, and p_1 ranges over the higher part of s_1, and also depends on p_2 ranging over the lower part of s_2, then the converse is true, i.e. that a parameter p_3 ranging on the higher part of s_2 depends on a parameter p_4 ranging on the lower part of s_1.
Hence, different trade-offs denote good/bad quality according to which criterion is preferred. *oqval* formalizes the observation that quality parameters are defined according to some *principle*, e.g. in the example, "high" parameters could be defined with reference to a *transparency* principle, while the "low" parameters could be defined with reference to a *computational efficiency* principle.
When combining principles, the need for a trade-off typically arises, producing either a *preference ordering function*, or a *relaxation of parameters*.
In our example project, the preference ordering function is:

$$pref(p_1^q, p_2^r, p_1^q, p_2^r, c) \mapsto p_i^x$$

where *q a* and *r* are principles, p_i^x is a parameter defined by a principle, and *c* is a local constraint (a "meta-parameter"), e.g. *availability of resources*, *user overruling*, *good practice*, etc.
If no local constraint can be applied to create a preference ordering function, the trade-off can resort to a relaxation of parameters. In our example project, either high density can be relaxed to e.g. *medium-high density*, or low complexity can be relaxed to e.g. *medium-low complexity*.

c) Which examples? There are typical examples and patterns of good/bad quality for each measure. This version of the technical report does not include a complete set of examples, and even less of patterns. In the next version, more examples will be provided after the analysis of a sample set of ontologies. In future versions, we'll

propose patterns of good/bad quality based on correlation between success stories, user satisfaction feedback, and measures.

2.2 Measuring the structural dimension

The structural dimension of ontologies focuses on syntax (e.g. graph structure), and and formal semantics.
Here we propose our own treatment of structural dimensions. The idea is to define a general function like the following:

$$M = \langle D,S,mp,c \rangle$$

<D>=Dimension is the graph property or concept we want to measure: the intensional counterpart of the metric space.
<S>=Set of graph elements is the collection of elements in the graph (which may be seen as the ontology structure).
<mp>=Measurement procedure is the procedure executed to perform the measurement.
<c>=Coefficient of measurement error adjusts for context-related variations on measurement procedure.

The value of the function is a real number obtained by applying a measurement procedure *mp* for a dimension *D* to a set *S* of graph elements, modulo a coefficient *c* (if any), i.e.:

$$mp_{D,c,S} \xrightarrow[yields]{} m \in \Re$$

The usual measuring procedure is *counting*, i.e. a function that relates a set of elements $x \in S$ to natural numbers. Sometimes a non-trivial algorithm is necessary to perform counting.
Within the possible sets of graph elements, we'll consider in particular the following sets:

- The set of graph nodes *G* from a graph *g*, $G \subseteq S$
- The set of root nodes $ROO \subseteq G$, where the root nodes are those having no outgoing *isa* arcs in a graph *g*.
- The set of leaf nodes $LEA \subseteq G$, where the leaf nodes are those having no ingoing *isa* arcs in a graph *g*.
- The sets of sibling nodes $SIB_{j \in G}$ connected to a same node *j* in a graph *g* through *isa* arcs.
- The set of paths *P* where $\forall j \in P \Rightarrow j \subseteq G$, where a path *j* is any sequence of directly connected nodes in a digraph *g* starting from a root node $x_{\in ROO}$ and ending at a leaf node $y_{\in LEA}$.
- The set of levels ("generations") *L* where $\forall j \in L \Rightarrow j \subseteq G$, where a generation *j* is the set of all sibling node sets having the same distance from (one of) the root node(s) $r_{\in ROO}$ of a digraph *g*.
- The sets of graph nodes $N_{j \in P}$ from a same path *j* in a digraph *g*
- The sets of graph nodes $N_{j \in L}$ from a same generation *j* in a digraph *g*

Measures for depth

Depth is a graph property related to the cardinality of paths in a graph, where the arcs considered here are only isa arcs. This measure type only applies to digraphs (directed graphs). We distinguish the following depth measures.

(M1) **Absolute depth**:

$$m = \sum_{j}^{P} N_{j \in P}$$

where $N_{j \in P}$ is the cardinality of each path j from the set of paths P in a graph g.

(M2) **Average depth**:

$$m = \frac{1}{n_{P \subseteq g}} \sum_{j}^{P} N_{j \in P}$$

where $N_{j \in P}$ is the cardinality of each path j from the set of paths P in a graph g, and $n_{P \subseteq g}$ is the cardinality of P.

(M3) **Maximal depth**:

$$m = N_{j \in P}$$
$$\forall i \exists j (N_{j \in P} \geq N_{i \in P})$$

where $N_{j \in P}$ and $N_{i \in P}$ are the cardinalities of any path i or j from the set of paths P in a graph g.

Measures for breadth

Breadth is a property related to the cardinality of levels ("generations") in a graph, where the arcs considered here are again only isa arcs. This measure only applies to digraphs. We distinguish the following breadth measures.

(M4) **Absolute breadth**:

$$m = \sum_{j}^{L} N_{j \in L}$$

where $N_{j \in L}$ is the cardinality of each generation j from the set of generations L in the digraph g.

(M5) **Average breadth**:

$$m = \frac{1}{n_{L \subseteq g}} \sum_{j}^{L} N_{j \in L}$$

where $N_{j \in L}$ is the cardinality of each generation j from the set of generations L in a digraph g, and $n_{L \subseteq g}$ is the cardinality of L.

(M6) Measure(**maximal breadth**, set of graph elements, counting, c) =

$$m = N_{j \in L}$$
$$\forall i \exists j (N_{j \in L} \geq N_{i \in L})$$

where $N_{j \in L}$ and $N_{i \in L}$ are the cardinalities of any generation i or j from the set of generations L in a graph g.

Examples:

Measures for tangledness

Tangledness is related to the multihierarchical nodes of a graph, where the arcs considered here are again only isa arcs. This measure only applies to digraphs.

(M7) **Tangledness**:

$$m = \frac{n_G}{t_{\in G \wedge \exists a_1, a_2 (isa(m,a_1) \wedge (isa(m,a_2)}}$$

where n_G is the cardinality of G, and $t_{\in G \wedge \exists a_1, a_2 (isa(m,a_1) \wedge (isa(m,a_2)}$ is the cardinality of the set of nodes with more than one ingoing *isa* arc in g.

Example:

Fig.5 shows some generic measures on an ontology graph.

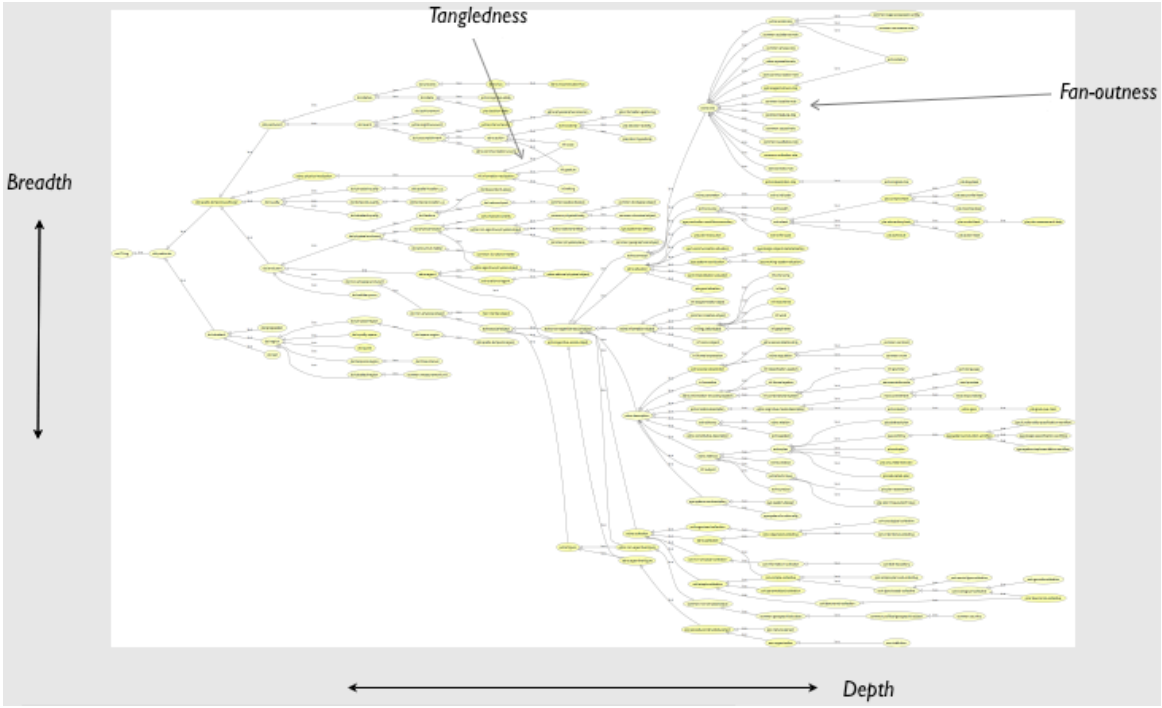

Figure 5. Examples of measures over an ontology graph (http://dolce.semanticweb.org). The root node is here drawn leftmost.

Measures for fan-outness

Fan-outness is related to the "dispersion" of graph nodes, where the arcs considered here are isa arcs. We distinguish the fan-outness measures related to *leaf node sets*, and fan-outness measures related to *sibling node sets* ("internal dispersion").

(M8) **Absolute leaf cardinality**:

$$m = n_{LEA \subseteq g}$$

where $n_{LEA \subseteq g}$ is the cardinality of the set *LEA* in the digraph *g*.

(M9) **Ratio of leaf fan-outness**:

$$m = \frac{n_{LEA \subseteq g}}{n_G}$$

where $n_{LEA \subseteq g}$ is the cardinality of the set *LEA* in the digraph *g*, and n_G is the cardinality of *G*.

(M10) **Weighted ratio of leaf fan-outness**:

$$m = \frac{n_{LEA \subseteq g}}{\sum_{j}^{P} N_{j \in P}}$$

where $n_{LEA \subseteq g}$ is the cardinality of the set *LEA* in the digraph *g*, and $\sum_{j}^{P} N_{j \in P}$ is the absolute depth measure for *g*.

(M11) **Maximal leaf fan-outness**:

$$m = N_{j \in SIB}^{j \subseteq LEA}$$
$$\forall i \exists j (N_{j \in SIB}^{j \subseteq LEA} \geq N_{i \in SIB}^{i \subseteq LEA})$$

where $N_{j \in SIB}$ and $N_{i \in SIB}$ are the cardinalities of any sibling set *i* or *j* of leaf nodes, from the set of sibling sets *SIB* in a graph *g*.

(M12) **Absolute sibling cardinality**:

$$m = \sum_{j}^{SIB} N_{j \in SIB}$$

where $N_{j \in SIB}$ is the cardinality of a sibling set *j* from *SIB* in the graph *g*.

(M13) **Ratio of sibling fan-outness**:

$$m = \frac{\sum_{j}^{SIB} N_{j \in SIB}}{n_G}$$

where $\sum_{j}^{SIB} N_{j \in SIB}$ is the absolute sibling cardinality for the digraph *g*, and n_G is the cardinality of *G*.

(M14) **Weighted ratio of sibling fan-outness**:

$$m = \frac{\sum_{j}^{SIB} N_{j \in SIB}}{\sum_{j}^{P} N_{j \in P}}$$

where $\sum_{j}^{SIB} N_{j \in SIB}$ is the absolute sibling cardinality for the digraph *g*, and $\sum_{j}^{P} N_{j \in P}$ is the absolute depth measure for *g*.

(M15) **Average sibling fan-outness**:

$$m = \frac{\sum_{j}^{SIB} N_{j \in SIB}}{n_{SIB}}$$

where $\sum_{j}^{SIB} N_{j \in SIB}$ is the absolute sibling cardinality for the digraph *g*, and n_{SIB} is the cardinality of the set *SIB* for *g*.

(M16) **Maximal sibling fan-outness**:

$$m = N_{j \in SIB}$$
$$\forall i \exists j (N_{j \in SIB} \geq N_{i \in SIB})$$

where $N_{j \in SIB}$ and $N_{i \in SIB}$ are the cardinalities of any sibling set *i* or *j* from the set of sibling sets *SIB* in a graph *g*.

Some fan-outness measures can be provided for assessing the kind of sibling sets in an ontology. In particular, we are interested in two kinds: sibling sets that are based on a *metric space*, and sibling sets that are *lists of values* (but not from a metric space).
These two kinds are interesting for quality assessment because they are often counter-examples to the "badness" of a high fan-ouness. On the other hand, there can be counter-counter-examples when a design pattern is assumed which suggests that values from metric spaces and lists have to be represented as individuals, and not classes. In this case, in fact, those value sets are no more sibling sets, because the *instanceOf* arc applies to them.

(M17) **Average sibling fan-outness without metric space**:

$$m = \frac{\sum_{j}^{SIB} N_{j \in SIB-MS}}{n_{SIB-MS}}$$

where $\sum_{j}^{SIB} N_{j \in SIB-MS}$ is the absolute sibling cardinality (less metric spaces) for the digraph g, and n_{SIB-MS} is the cardinality of the set *SIB* for g, less the sets of metric values.

(M18) **Average sibling fan-outness without lists of values**:

$$m = \frac{\sum_{j}^{SIB} N_{j \in SIB-LV}}{n_{SIB-LV}}$$

where $\sum_{j}^{SIB} N_{j \in SIB-LV}$ is the absolute sibling cardinality (less lists of values) for the digraph g, and n_{SIB-LV} is the cardinality of the set *SIB* for g, less the lists of values.

Measure M17 and M18 are meaningful only when metric values or lists of values are represented as classes. For metric spaces this is rarely practiced, specially becaue languages like OWL have separate domains for datatypes.

Measures for *differentia specifica*

Differentia specifica (Latin for "specific difference", the provenance of this expression going back to Aristotle) is related to the "rationale" behind sibling node sets. The rationale behind a sibling node set can be measured by looking for arcs that do not represent isa relationships, and are shared by all siblings in the set (these arcs represent a common relational property).
For a more relevant measure, we exclude from this measure the sibling nodes that represent values from a metric space of just a list (see (M17-18). We distinguish the following specific difference measures.

(M20) **Ratio of sibling nodes featuring a shared *differentia specifica***:

$$m = \frac{\sum_{j}^{SIB} N_{j \in SIB, \forall x_{x \in j} \exists \rho, \varphi(\rho(x,y) \wedge \varphi(y))}}{\sum_{j}^{SIB} N_{j \in SIB}}$$

where $\sum_{j}^{SIB} N_{j \in SIB, \forall x_{x \in j} \exists \rho, \varphi(\rho(x,y) \wedge \varphi(y))}$ is the absolute cardinality of siblings sharing a common relational property ρ for each sibling set j for the digraph g, and $\sum_{j}^{SIB} N_{j \in SIB}$ is the absolute cardinality of siblings for g.

(M21) **Ratio of sibling sets featuring a shared *differentia specifica* among elements**:

$$m = \frac{n_{SIB(DF)}}{n_{SIB}}$$

where $n_{SIB(DF)}$ is the cardinality of the set *SIB(DF)* including only the sibling sets whose elements share a specific difference. More precisely, an element $x \in j$ (a sibling set from *SIB(DF)*) must have a same relational property ρ holding for different values from a same class φ; formally, for a sibling set j: $\forall x \in j \ \exists \rho, \varphi(\rho(x,y) \wedge \varphi(y))$. n_{SIB} is the cardinality of the set *SIB* for g.

Measures for density

Density can be defined as the presence of clusters of classes with many non-taxonomical relations holding among them (wrt to overall ontology graph). For example, so-called *core ontology patterns* (for thematic roles in events, contracts, diagnoses, etc.) usually constitute dense areas in an ontology. The following measures can be established.

(1) Various clustering techniques can be used to detect dense areas, and the absolute size and number of them can be measured.
(2) A measure of the relevance of those areas for the overall ontology can be obtained by calculating the proportion of classes and properties in the ontology, which logically depend on the dense areas.
(3) Dense areas can be -explicitly or implicitly- a specialization of *ontology content patterns* [Gangemi 2005].

Measures for modularity

A *module* is any subgraph *sg* of a graph *g*, where the set of graph elements *S'* for *sg* is such that $S' \subseteq S$.
Two modules sg_1 and sg_2 are *disjoint* when only ≥ 0 *isa* arcs a_i connect sg_1 to sg_2, and each a_i has the same direction.

Modularity is related to the asserted modules of a graph, where the arcs considered here are either *isa* or *non-isa* arcs. We distinguish the following modularity measures.
The set of modules from a graph *g* is called here *M*.

(M22) **Modularity rate**:

$$m = \frac{n_M}{n_S}$$

where n_M is the cardinality of *M*, and n_S is the cardinality of *S* (the set of graph elements).

Example:

(M23) **Module overlapping rate**:

$$m = \frac{\sum_n n_{uoap} - \sum_n n_{uoadp}}{n_{\{sg_1, sg_2\}} - n_{\{sg_1 | sg_2\}}}$$

where $\sum_n n_{uoap}$ is the sum of the cardinalities of the sets of *uniquely directed* arcs between the members of each module pair *p* from *M*; $\sum_n n_{uoadp}$ is the sum of the cardinalities of the sets of *uniquely directed* arcs between the members of each disjoint module pair *p* from *M*; $n_{\{sg_1, sg_2\}}$ is the cardinality of the set of module pairs, and $n_{\{sg_1 | sg_2\}}$ is the cardinality of the set of disjoint module pairs.

Given an appropriate procedure, it's possible to create a modularization of a non-modularized graph.

Measures for logical adequacy

Logical adequacy is related to graphs having a formal semantics, where the arcs considered here are either isa or conceptual relation arcs. We distinguish the following logical adequacy measures.

(M24) **Consistency ratio**:

$$m = \frac{n_{Cons}}{n_G}$$

where n_{Inc} is the cardinality of the set of consistent classes from *g*, and n_G is the cardinality of the set of (class) nodes from *g*.

Example

(M25) **Generic complexity**: a complexity scale, e.g. the one used for description logics

Example

Cf. D. Calvanese. Data Complexity of Query Answering in Description Logics, http://www.inf.unibz.it/~calvanese/papers-html/DL-2005.html.
A useful tool to automatically check the complexity of OWL ontologies is SWOOP (http://www.umbc.edu).
More specific measures can be made on particular constructs that affect the actual computational time. We consider here *anonymous classes*, *cycles*, and *inverse properties*.

(M26) **Anonymous classes ratio**:

$$m = \frac{n_{Anon}}{n_G}$$

where n_{Anon} is the cardinality of the set of anonymous classes, and n_G is the cardinality of the set of (class) nodes from *g*.

(M27) **Cycle ratio**:

$$m = \frac{\sum_{k}^{P} N_{k \in P}}{\sum_{j}^{P} N_{j \in P}}$$

where $\sum_{k}^{P} N_{k \in P}$ is the absolute depth measure for the set of cyclic paths $k_{1 \ldots n}$, $k_i \in P$ in the digraph *g*, and $\sum_{j}^{P} N_{j \in P}$ is the absolute depth measure for *g*.

(M28) **Inverse relations ratio**:

$$m = \frac{n_{InvR}}{n_R}$$

where n_{InvR} is the cardinality of the set of inverse relations represents by arcs in *g*, and n_R is the cardinality of the set of relations represented by arcs in *g*.

Other measures related to logical constructs are relevant for ontology evaluation, in particular we treat here *class/relation ratio* and *class/axiom ratio*.

(M29) **Class/relation ratio**:

$$m = \frac{n_{G \in S}}{n_{R \in S}}$$

where $n_{G \in S}$is the cardinality of the set of classes represents by nodes in *g*, and $n_{R \in S}$ is the cardinality of the set of relations represented by arcs in *g*.

(M30) **Axiom/class ratio**:

$$m = \frac{n_{A \in S}}{n_{G \in S}}$$

where $n_{G \in S}$is the cardinality of the set of classes represents by nodes in *g*, and $n_{A \in S}$ is the cardinality of the set of axioms represented by subgraphs in *g*.

(M31) **Individual/class ratio**:

$$m = \frac{n_{I \in S}}{n_{GI \in S}}$$

where $n_{G \in S}$is the cardinality of the set of classes represents by nodes in *g*, and $n_{I \in S}$ is the cardinality of the set of individuals represented by special nodes in *g*.

Presence of a reification vocabulary.
The presence of individuals in an ontology is usually not very frequent, and limited to cases where nominals are used in axioms, e.g.

Italian(x) $=_{df}$ Citizen(x) $\wedge$ $\exists y$(bornIn(x,y) $\wedge$ y=*Italy*)

On the other hand, the individual/class ratio can be higher in presence of a reification vocabulary. This is typically the case e.g. when classes are to be used as values in an axiom, for example:

NurseGuideline(x) $=_{df}$ Guideline(x) $\wedge$ $\exists y$(hasTarget(x,y) $\wedge$ y=*MaximalNurseCollective*)

In this example, *MaximalNurseCollective* is an individual used to reify the collective of all persons having the role of *nurse*, and such individuals are ideally asserted as instances of the class *Collective*, which can be defined in an appropriate vocabulary intended for representing that kind of reified entities.

Measures for meta-logical adequacy

Formal semantics can go beyond first-order, in order to attempt at representing functional adequacy as well (cf. 2.3).
The major example of that attempt is *OntoClean* [Guarino&Welty 2004]. OntoClean aims to classify the classes of an ontology according to some meta-properties, e.g. *rigidity*, *unity*, *dependence*. OntoClean metaproperties try to reduce some functional measures to the measurement of adequacy wrt series of possible worlds (states of affairs). For example:

- stability of a property across a series of temporal states for a same entity (*rigidity*) e.g. *person* vs. *student*
- disjointness of sets of properties across a series of (different) topological states for a same entity or cluster (*unity*) e.g. *dog* vs. *rubbish*
- stability of a property across a series of states featuring different relational properties for a same entity (*dependence*) e.g. *dog* vs. *dogtail*

As we explain in 2.3, possible worlds can be used to check the quality of an ontology only in an *ex-post* way, e.g. by analyzing the history of a temporal database built according to the ontology that must be evaluated. This is unfit, since an ontology is supposed to be evaluated before its application, not afterwords.
For this reason, OntoClean methodology suggests designers or experts to assign meta-properties in advance, and then to use these assignments to check the meta-logical consistency of the taxonomy. For example, the principle of *meta-level integrity* (cf. 2.5) requires parameters for *meta-consistency*, e.g. a parameter by which no *rigid* class can be subsumed by an *anti-rigid* one, because a (temporally) stable class would result to depend on an unstable one.

(M32) **Meta-consistency ratio**:

$$m = \frac{n_{MCons}}{n_G}$$

where n_{MCon} is the cardinality of the set of meta-consistent classes from *g*, and n_G is the cardinality of the set of (class) nodes from *g*.

Measures for degree distribution

There is a number of statistical-analytical notions that are commonly applied to the analysis of graphs. For instance, the application of statistical analysis to a graph allows to isolate patterns in the form of dense areas. These areas may be characterized through the following notions:

Degree distribution, which measures the probability of a vertex to have a certain degree (i.e. the sum of its out- and in- degrees). When the probability of a vertex having a degree k (P(k)) follows a power-law distribution (P(k) ≈ k- r) -- as opposed to a Poisson distribution -- it is possible to conclude that the structure of (the system represented by) the graph is not random. Moreover, power law distributions are characterized by the γ

exponent and are called **scale-free networks**. In other words, they show the same properties independently of the scale at which they are observed.

Small world: a graph is a small world if the a**verage minimum path length d** between vertices is short, usually scaling logarithmically with the total number of vertices. Graphs showing an average path length similar to random graphs of the same size and average degree are very likely small worlds, $d \approx d_{random}$.

Clustering coefficient: It measures the probability that two neighbors of a given node are also neighbors of one another. For random graphs it is a small quantity. However, CSs show a high clustering compared to random graphs, $C >> C_{random}$. A high clustering confirms small-worldness.

Now, the degree distribution might support the structural measuring of various types of systems, among which ontologies [3]. In particular, if graphs are used for modeling systems that developed over time without a central control (like the World Wide Web or large-scale collaborative ontologies) nodes (i.e. web-pages, respectively, concepts) with a high degree have probably become crucial to the very existence of the entire structure, and in the case of ontologies this possibly means that they are semantically crucial. On the other hand, the γ exponent may be taken as a sign of the "recursive" structure of (the system represented by) the graph. Similar considerations hold for small worlds and for the clustering coefficient.
Degree distribution measures can be combined with measures related to dense areas.

2.3 Measuring the functional dimension

The functional dimension is coincident with the main purpose of an ontology, i.e. specifying a given conceptualization, or a set of contextual assumptions about a world. Such specifications, however, are always approximate, since the relationship between an ontology and a conceptualization is never straightforward (Fig.2). Hence, an appropriate evaluation strategy should involve a measurement of the degree of such approximation. In the semiotic view of ontologies, this amounts to measuring the extent to which a (syntactic) graph expresses a context-bound intended meaning, possibly with respect to a formal semantic interpretation.[4]
The problem, hence, is to find ways of measuring the extent to which an ontology mirrors a given expertise, or competency: something that is "in the experience" of a given community and that includes not only a corpus of documents, but also theories, practices and know-hows that are not necessarily represented in their entirety in the available documents. This seems to imply that no automatized method will ever suffice to the task and that intellectual judgement will always be needed However, both automatic and semi-automatic techniques can be applied that make such evaluation easier, less subjective, more complete and faster (cf. [Daelemans et al. 2004]).

2.3.1 Formalizing functional measures

[3] http://dmag.upf.es/livingsw/
[4] We remind here that we are considering also ontologies that are not given a formal interpretation, such as terminologies or thesauri.

We propose here some functional measures that are variants of the basic measures introduced by [Guarino 2004], which uses an analogy with *precision* and *recall* measures (cf. 2.3.3 for the definition in its original context). Precision and recall are measures which are widely used in information retrieval (cf. [Baeza-Yates & Ribeiro-Neto,1999]) and are defined as follows (TP=True Positive; FP=False Positive; FN=False Negative):

Precision:

$$P = \frac{n_{TP}}{n_{TP} + n_{FP}}$$

where n_{TP} is the cardinality of the set of true positives, and n_{FP} is the cardinality of the set of false positives.

Recall:

$$R = \frac{n_{TP}}{n_{TP} + n_{FN}}$$

In the context of ontology evaluation, the definition is adapted by choosing an appropriate domain for the positives resp. negatives from the matching between the ontology structure and the intended usage and meaning.
In particular, [Guarino 2004] proposes a possible-world semantics to characterize that matching. Given a logical language *L* that implicitly commits to a *conceptualization C*, an ontology's purpose is to capture all and only those models of *L* that are compatible with *C*.
These models are called the *intended models* $I_k(L)$, *k* being the commitment to a certain intepretation *I* for *L*. In this semantics, an ontology *O* using *L* is «a logical theory designed in such a way that the set $O_k(L)$ of its models relative to *C* under the commitment *k* is a suitable approximation of the set $I_k(L)$».
Given a conceptualization: $C = \langle \Delta, W, R \rangle$, where Δ is a set of relevant entities, *W* a set of possible worlds, and *R* a set of intensional relations[5], precision and recall are defined as follows (Fig. 6):

O_Precision:

$$OP = \frac{n_{TP}^{O_k(L)}}{n_{TP}^{O_k(L)} + n_{FP}^{O_k(L)}}$$

i.e. the proportion of intended models $_{TP}^{O_k(L)} \subseteq I_k(L)$ (True Positives) over Δ, on the sum of all *O* models $O_k(L)$, which can include False Positives $_{FP}^{O_k(L)} \not\subset I_k(L)$.

O_Recall (renamed *coverage* by [Guarino 2004]):

[5] An intensional relation is a function from *W* to the set 2^{D^n} of all possible n-ary extensional relations on *D*.

$$OR = \frac{n_{TP}^{O_k(L)}}{n_{TP}^{O_k(L)} + n_{FN}^{O_k(L)}}$$

i.e. the proportion of intended models $_{TP}^{O_k(L)} \subseteq I_k(L)$ (True Positives) over Δ, on the sum of all intended models $I_k(L)$, which can include False Negatives $_{FN}^{O_k(L)} \subseteq I_k(L)$.
In [Guarino 2004] the formulas for precision and covergae use directly I_k as a given set, and does not use positives and negatives; for example, the precision formula is: $\frac{n^{O_k(L)} \cap n^{I_k(L)}}{n^{O_k(L)}}$. But in realistic ontology projects, I_k is not a given set: we are only able to detect -indirectly at best- (a sample of) false positives and false negatives. For this reason, we prefer to maintain the analogy with IR, which is not based on the assumption that the expected set of results is given in advance.

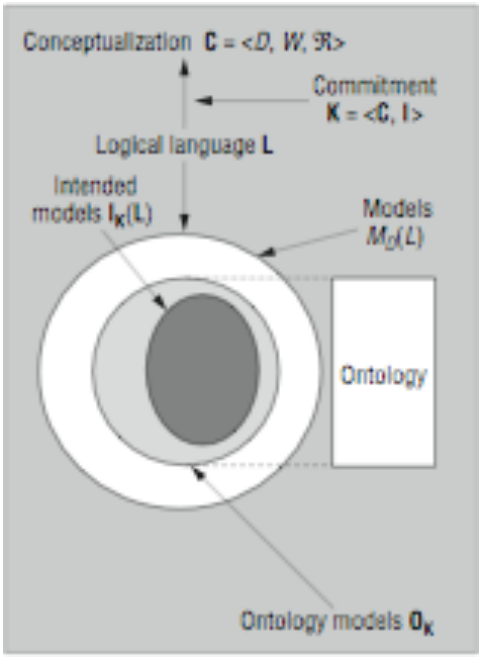

Figure 6. The relationship between an ontology and a conceptualization.

In other words, an ontology can accept unintended models, resulting in lower precision, or can miss intended models, resulting in lower recall. Fig.7 is a picture of the possible cases resulting from this definition of precision and recall.

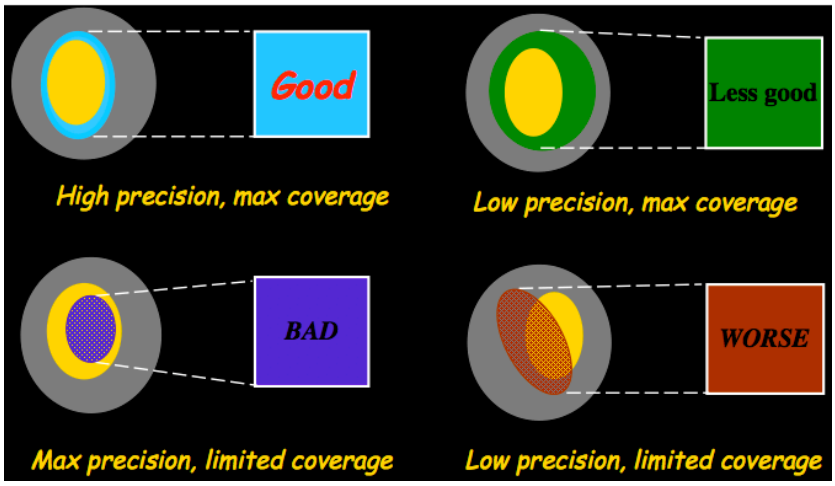

Figure 7. Precision and recall (coverage) of an ontology. The grey ovals includes all models allowed by the logical language. The yellow ovals include the intended models. The oval projections of rectangular spaces include the models allowed by an ontology. 100% precision implies that all non-intended models are excluded; 100% coverage implies that all intended models are included.

But this is not the whole story, since models can map different states of affairs. Usually, an intended conceptualization "prospects" a set of possible states of affairs, which can contain *possible* distinctions that are not expressible in an ontology (cf. the *BWO* example in the next section). This is a direct consequence of the mismatch between cognitive and formal semantics (see section 1.). Such mismatch results in a decrease of *factual precision* and *coverage* while model-based precision and coverage remain stable.
Factual precision/coverage is tentatively formalized through the notion of *accuracy* (inspired by [Guarino 2004] that does not provide a formula for it), which tries to measure the fitness to an intended conceptualization $C = \langle \Delta, W, R \rangle$ by mapping states of affairs to possible worlds in W:

O_Accuracy:

$$OA = \frac{n_{TP}^{O_k(L)_W}}{n_{TP}^{O_k(L)_W} + n_{FP}^{O_k(L)_W}} \cdot \frac{n_{TP}^{O_k(L)_W}}{n_{TP}^{O_k(L)_W} + n_{FN}^{O_k(L)_W}}$$

i.e. the proportion of intended states of affairs ${}^{O_k(L)_W}_{TP} \subseteq I_k(L)_W$ (True Positives) over the sum of all O states of affairs $O_k(L)_W$, which can include False Positives ${}^{O_k(L)}_{FP} \not\subseteq I_k(L)$, multiplied by the proportion of intended states of affairs ${}^{O_k(L)_W}_{TP} \subseteq I_k(L)_W$ (True Positives) over the sum of all intended states of affairs $I_k(L)_W$, which can include False Negatives ${}^{O_k(L)}_{FN} \subseteq I_k(L)$.

An example: *BWO*

We consider a simple axiomatic theory: the Blocks World Ontology (*BWO*), including:
A signature: {On, Block}
A set of axioms:
(A1) $On(x,y) \rightarrow Block(x) \wedge Block(y)$
(A2) $On(x,y) \rightarrow \neg On(y,x)$ *(antisymmetry)*

(A3) (On(x,y) ∧ On(y,z)) → On(x,z) *(transitivity)*

We could now consider the following model *M* for *BWO*, consisting of a set of propositions:
(P1) On(*red_block#1, blue_block#1*)
(P2) On(*green_block#1, red_block#1*)
(P3) On(*yellow_block#1, red_block#1*)

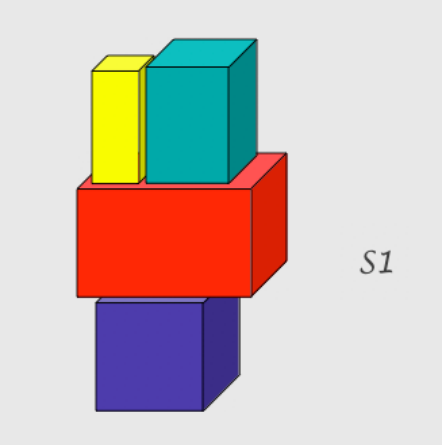

Fig.8. A typical state of affairs for M in BWO.

A typical intended state of affairs *S1* (representable as a possible world) for *M* is depicted in Fig.8: the red block is on the blue block, and the green and yellow blocks are on the red one.
But how to be sure that there are no other states of affairs that are not intended, which can be compatible with *M*? And that there are no intended states of affairs that are not captured by *M*?
In other words, does *BWO* catch all and only the intended meaning of the agent that defines or uses *BWO*? Also: how to know about that intended meaning? What are its boundaries? We will propose later that the *intended usage* must be considered firstly.
We show with a further state of affairs that these questions deserve a non-trivial answer. Let's consider a less typical state of affairs *S2* for *M* (Fig.9).

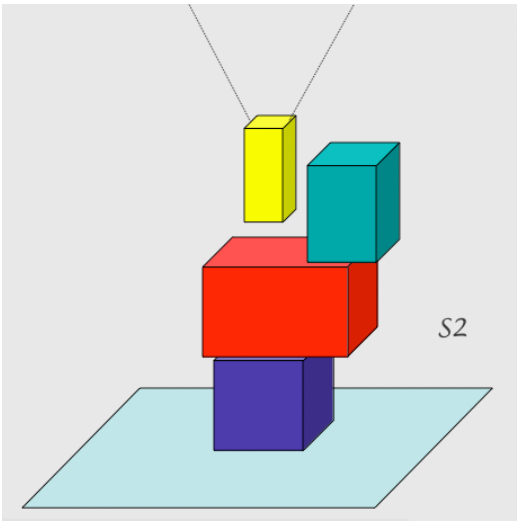

Fig.9. A less typical state of affairs for M in BWO.

S2 appears as less typical because *S1* is closer to the conventions applied in classic AI and computer science examples, nonetheless *S2* can even be more realistic: a plane surface is depicted which intuitively bears the weight of the blocks; a block (the green one) can lie on the red one without being strictly "on" the blue (it lacks a vertical alignment); the yellow block can hang through a pair of cables, without actually touching the red block, so being *over* it, but still "on" it, if *M* is the best approximation to *S2* in *BWO* (since it admits a "disconnected" on).
Therefore, *S2* requires a richer theory to be distinguished from *S1*. This means that either (i) *S2* is an intended state of affairs (it is expected by the intended conceptualization), but *BWO* is not accurate enough to distinguish it from others accounted by the same model (there is a false negative possible world); or (ii) *S2* is not an intended state of affairs, but *BWO* is not accurate enough to exclude it (there is a false positive possible world).
In both cases, the formal semantics does not comply with the intended conceptualization (the cognitive semantics, cf. 1.).
Additional aspects may be lacking for some intended usage, e.g.: temporal and spatial constraints; the kind of space to be assumed; the color qualities of the blocks, etc.

Precision, recall and accuracy have been adapted to formalize the functional fitness of an ontology. On the other hand, the purely formal characterization of P/R and accuracy is neutral with respect to what procedure is used to obtain the (false/true) positives resp. negatives.
But such a procedure is needed, and it greatly influences what data can be obtained. Which models are intended? which states of affairs? are intended models decided with reference to intended states of affairs? or are intended states of affairs only a refinement of intended models?
In realistic cases, sometimes an ontology is taken as a prescriptive set of rules, sometimes it is received as a guideline, and sometimes it is something to discuss or modify. These different attitudes change the scenario in which we are supposed to answer those questions. Unfortunately, the formalization does not help much, and can even be misleading without an explicit procedure.
Simply stated, the *applicability* of functional measurement is key to the functional evaluation of ontologies.

Applicability of functional measurement

The applicability of functional measurement is based on a process of *matching*. While structural measurement analyses one structure (a graph), functional measurement analyses the correspondence between two structures: a graph, and something else, where this something else is extremely difficult to capture without being part of the application context.
Formal semantics can help defining a task by providing an *intepretation* to a graph, but such interpretation depends on the commitment of (one or more) agent(s), and such commitment is motivated by a typical *expertise* for a *task* in that context.

Intended conceptualization as (a schema of) expertise for some task

We are proposing that an intended conceptualization corresponds to (part of) the expertise of ontology intended users, where the expertise' boundary is provided by the task that should be accomplished with the help of the ontology.
There is some disagreement on if expertise should be encoded in ontologies, or in knowledge bases [Guarino and Giaretta 1995]. Anyway, since a knowledge base depends on a schema (i.e. an ontology), at least the *schema of an expertise* should be encoded as

an ontology. This is the point we are making here: states of affairs can be prospected (are intended) only wrt to expertise for a task, then an ontology should be aimed at capturing at least the schema of that expertise.
Another consequence of our proposal is that models are not "intended": they can be *admitted* or not by the users' conceptualization (cf. Fig.2), on the basis of said expertise. In other words, our semiotic perspective turns the problem upside-down: formalization comes as a tool to represent expertise and task, not as a requirement independent from expertise and task.
Expertise and tasks need to be captured where they are enacted or deposited, then formalized.
But, since expertise is by default in the cognitive "black-box" of rational agents, ontology engineers have to elicit it from the agents, or they can assume a set of data as a *qualified expression* of expertise and task, e.g. texts, pictures, diagrams, db records, terminologies, metadata schemas.

2.3.2 Qualified expressions of intended conceptualization: some measurement methods

Based on these assumptions, precision, recall and accuracy of an ontology graph can be measured against: a) experts' judgment, or b) a data set assumed as a qualified expression of experts' judgment. Therefore, we distinguish between *black-box* and *glass-box* measurement methods:

(1) Agreement assessment (*black-box*)
(2) User-satisfaction assessment (*black-box*)
(3) Task assessment: what has to be supported by an ontology? (*glass-box*)
(4) Topic assessment: what are the boundaries of the knowledge domain addressed by an ontology? (*glass-box*)
(5) Modularity assessment: what are the building blocks for the design of an ontology? (*glass-box*)

Black-box methods require rational agents, because they don't explicitly use knowledge of the internal structure of an expertise.
Glass-box methods require a data set that "represents" that knowledge, and, on this basis, we can treat the internal structure of those data *as if* it is the internal structure of an expertise.

Agreement assessment

When experts' judgment is taken into account, precision, recall and accuracy can only be measured through the proportion of *agreement* that experts have with respect to ontology elements; when a group of experts is considered, we may want to measure the *consensus* reached by the group's members.
Agreement assessment is a black-box measurement, since intended conceptualization is left in the experts' mind, and we only measure their approval (or the proportion of consensus) on the set of ontology elements. This makes the measurement very reliable at design-time, while it needs a reassessment at reuse-time.
Agreement assessment requires organized experts and their availability to take part in a trial. Many rhetorical and argumentation issues come into place when trying to measure consensual weights; e.g. discussing by "examples and counterexamples" is a typical technique. When a consensus-reaching methodology (e.g. [Uren et al. 2004], [DILIGENT]) or a modular design (see below) has been used in the ontology lifecycle,

this provides a preliminary measure for the evaluation of an ontology at *design-time* (but it should be reassessed at *reuse-time*).

User-satisfaction assessment

A more "black-box*ish*" method is the measure of *user satisfaction*, which can be carried out by means of dedicated polls, or by means of provenance, popularity, and trust assessment.
User satisfaction provides an indirect measure of fitness to expertise and task, but requires a careful profiling procedure to establish the competency/subject area of the users involved in a poll, or in a network of trust.

Task assessment

A glass-box method type is based on the availability of data about the *task* intended for an ontology. Therefore, it deals with measuring an ontology according to its fitness to some goals, preconditions, postconditions, constraints, options, etc.
This makes the measurement very reliable at design-time, while it needs a reassessment at reuse-time.
Task assessment requires a task specification. Three approaches can be singled out: *service-, solution-,* and *task-based*. The first two specify the task indirectly, while the third is a direct specification.

(1) Service-specification-based. The process model of an application can be used to evaluate the P/R of an ontology. For example, if a service requires a certain I/O pattern, the ontology should provide a vocabulary and axioms to the data involved in the I/O process, such that an appropriate application schema can be built, and the I/O process provides the expected results. Service specification can be a practical method to functional evaluation (in fact, it has been traditionally used in *conceptual modelling*), but it could miss the relevant social aspects of expertise, since a task is not equivalent to an application: requirements can need more expressivity than that provided for computational service specification.
For example, an experiment has been carried out on the Oracle Human Resources (HR) schema (Fig.s 10,11) in the context of the EU WonderWeb project (http://wonderweb.semantiweb.org). The HR schema from the legacy application has been *reengineered* as an OWL ontology, showing just a few classes and properties (8/10) that actually refer to the HR domain, while more than 40 properties are provided just as foreign keys: they are mostly bound to the application requirement, not to the task requirements. After a careful *remodelling*, the number of classes and properties increased (15/16), but with no need for the additional properties for foreign keys. Incidentally, the remodelling changed the class/property ratio from approx. 1:6 to approx. 1:1, which is closer to ontologies usually considered as "accurate". This result is interesting here, because the remodelling process has considered HR from a social (organizational) perspective, not from a purely application-oriented perspective.
(2) Gold-standard-solution-based: A validated corpus of answers for a certain task (either computational or cognitive) can be used to evaluate the accuracy of an ontology. The corpus is then used as a *gold standard*. This method checks the performance of an ontology-driven system with reference to those answers (cf. [Porzel et al. 2004]). This method is more effective than (1), but requires an intellectual pre-processing of competency questions (see (3)) on a local basis, and therefore it is not easily generalizable.
(3) Explicit-task-based: Instead of matching an ontology to an application model or to a

pre-processed set of answers, we can directly match it to a *task specification*. A task specification must include *references to a domain ontology* (it combines with topic assessment, see next section). A task specification can be generic, then *usable to a large variety of applications and requirements*. A task specification should take into account the *social context* of prospective applications.

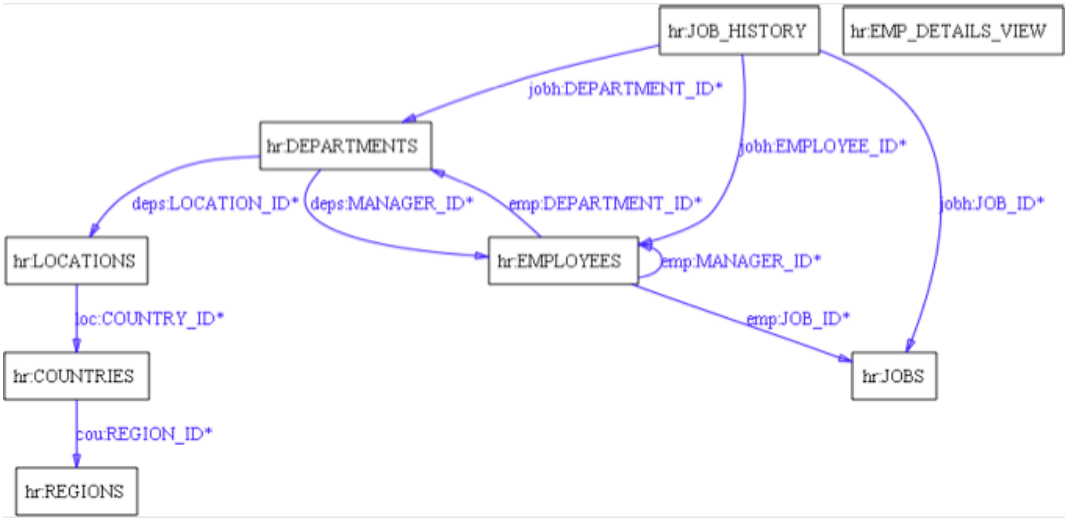

Figure 10. The Human Resources schema reengineering in OWL.

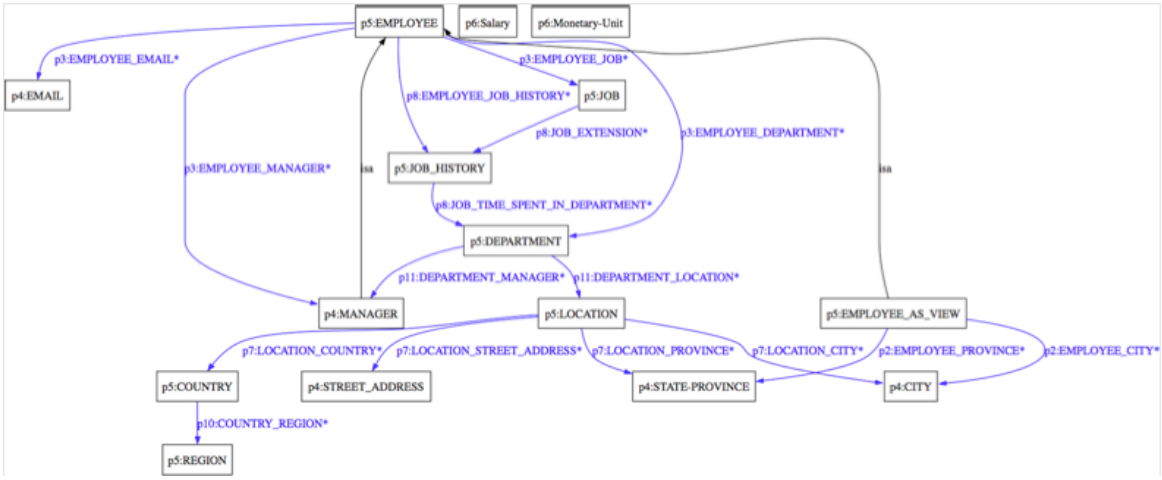

Figure 11. The Human Resources schema remodelled by means of a social ontology.

There are several examples of specification frameworks for tasks (cf. [Gangemi et al. 2004] for a review and an axiomatic theory of tasks and plans). We report here three very different approaches as a sample:

a) Competency questions [Grüninger et al. 1996] are an informal method to profile the type of queries that users may want to make to a knowledge base specified according to an ontology.
b) WSMO (Web Service Modelling Ontology) [Vasiliu et al. 2004] is a set of ontologies and specifications that can be used to formalize the profile, process-model, choreography, and orchestration of web services. WSMO specs can refer to a domain ontology and can be general.
c) COS (Core Ontology of Services) [Oberle et al. 2004] is an ontology that can be used to formalize the *social service* that motivates the implementation of a web service. COS specs are built *as* domain ontologies, can be very general,

and can use a built-in vocabulary for the social context of prospective applications.

Topic assessment

Another (complementary) non-black-box method type is based on the availability of data about the *topic* covered by an ontology.
It deals with measuring an ontology according to its fitness to an existing knowledge repository (an *information realization*, cf. section 1.). This makes the measurement reliable both at design-time, and at reuse-time.
Topic assessment requires a topic specification. Three approaches can be singled out. The first one is a *grey-box* approach, while the second and third are direct, glass-box approaches.

(1) Choose directory and annotate: operated on a subject directory by annotating the ontology with a subject label: it's a grey-box technique, because subject label do not actually reveal the internal structure of an expertise, but act like "placeholders" for it.
(2) Reengineer and match: operated on metadata repositories, such as terminologies, informal diagrams, DB and OO schemas, etc. Reengineering is based on best practices (e.g. thesauri to OWL), formal translations (e.g. FOL to OWL), and a lot of customization. It requires enrichment of ontology with information objects that can be matched against information realizations (e.g. lexical occurrences in texts). Once reengineered, a source can be either imported or mapped: the degree of difficulty in devising such activities provides P/R measures of the ontology. A more directed measure can be obtained by considering the source as a gold-standard-model (cf. [Maedche&Staab 2002]).
A special case of "reengineer and match" is the direct reuse of another ontology, without reengineering it.
(3) Extract and match: operated on data repositories, such as linguistic or image corpora, databases, etc., by extracting information patterns and matching them to ontology graph. Extraction is based mostly on learning techniques. Matching is controversial, because it depends on the way data are parsed; e.g. a same text can be parsed in different ways, thus obtaining different patterns. E.g. in NLP, parsing terms (how complex?) vs. syntactic structures. E.g. parsing statistically vs. rule-based. An experiment with statistical parsing of syntactic patterns is [Ciaramita et al. 2005]. Other work is in e.g. [Brewster et al. 2004]. See next section for a review.

The next section is dedicated to evaluation based on NLP techniques, which are currently the most investigated, and are combined with the other methods presented here, so that it deserves an independent presentation.

2.3.3 Evaluation with NLP

When the ontology is lexicalized; i.e., it defines, at least to some extent, what instances of classes and relations are called in natural language, and there exists a substantial amount of textual documents which contain information about the content of the ontology, Natural Language Processing (NLP) can support ontology evaluation in several ways. A typical such case is when the ontology directly supports information retrieval or text mining applications and thus concerns objects mentioned in web-pages or other large repositories of texts (e.g., newswire, biomedical or legal literature, etc.). One of the simplest examples of lexicalized ontology is the kind used for newswire information

extraction which is usually based on three classes: “person” (e.g., Mayor Giuliani, Kofi Annan, etc.), “location” (e.g., Houston, South East Asia, etc.), and “organization” (e.g., U.N., Enron, etc.). Sometimes these classes are also associated with relations such as “is-located-in” (e.g., is-located-in(Enron,Houston)) or “works-for” (e.g., works-for(Kofi_Annan,U.N.)).

Corpus-based ontology analysis

If there is a corpus of documents which contains the kind of information conceptualized in the ontology, NLP can be used to identify mentions of instances (i.e. occurrences in text) of classes and relations (which is more complicated than string matching, e.g., gene/protein names and relation/paraphrases identification) which are mentioned in the text. A corpus-based analysis of the ontology can reveal important properties of the ontology that might not be discovered otherwise. Most importantly it allows to estimate empirically the accuracy and the coverage of the ontology.

Distributional properties

By identifying mentions of ontological instances in the corpus it is possible to count the frequency of classes (similarly for relations). The relative frequency of each class c (or relation) is the proportion of mentions of ontology instances which are equal to c; i.e., P(c) = count(c)/sum_i count(c_i). The relative frequency measures the importance of each class and provides a first simple measure of the ontology quality. For example, in newswire text the three classes above have somewhat similar frequencies, while if the corpus analysis reveals that one of the classes is much more unlikely than the others this means that there is something wrong with the instances of that class. There might be errors or an insufficient number. If the ontology has a hierarchical, i.e., is-a, structure it is also possible to estimate the frequencies of higher of superordinate concepts, the frequency of a class c then would be the sum of the frequencies of its descendants. The probability of a concept in a hierarchy can be computed as P(c) = sum_j{c_j is descendant of c} count(c_j)/sum_i count(c_i). Each class can be seen as a random variable, this can be useful to estimate the information-theoretical measures such as entropy H(c) = -sum_j {c_j is descendant of c} P(c_j)logP(c_j). Entropy and other information theoretic measures can be used to identify classes that are particularly useful or “basic” (Gluck & Corter, 1985). Thus for example in a general purpose ontology, a concept such as “tree”, which has many descendants similar to each other, is likely more important than a concept such as “entity” which has very dissimilar descendants (e.g., organisms, artifacts, etc.).

One problem with trying to estimate distributional properties of the ontology directly is that the existing lexicon associated with the ontology might be insufficient because it contains only the names that the experts have listed. Notice that creating such “dictionaries” requires not only domain expertise but also lexicographic expertise and it is a slow and expensive process. Therefore typically the starting ontology lexicon is quite limited. This issue introduces two important metrics: precision and recall.

Precision and recall

When occurrences of the ontology instances are identified in a corpus two important measures come to play an important role in evaluating properties of the ontology: “precision” and “recall” (see previous section for a definition taken from [Baeza-Yates & Ribeiro-Neto,1999]). A few preliminary concepts need to be defined: a “true positive” (TP) is an instance which is correctly labeled with one class defined by the ontology; e.g., “Alan Greenspan” in the example below:

1)

Word	Guessed/True Label	Answer type
Fed	0/ORG	FN
chairman	0/0	TN
Alan	PER/PER	TP
Greenspan	PER/PER	TP
was	0/0	TN
in	0/0	TN
Philadelphia	LOC/LOC	TP
today	LOC/0	FP
.	0/0	TN

A "false positive" (FP) is an instance which is incorrectly labeled with a class label; e.g., "today" labeled as a "location" in Example (1). Similarly, a "false negative" (FN) is instance of a class which is not recognized as such, e.g. "Fed" in example (1), while a "true negative" (TN) is correctly not recognized as an instance of a class; e.g., "chairman", "was", etc. in Example (1).

Intuitively, precision measures the ability of a system in recognizing instances of a given class, while recall measures the coverage of the system, that is how many true instances were left out. Measuring precision and recall requires manual tagging of enough textual data to be able to compare the empirical lexicon so generated with the ontology lexicon. Typically, the lexicon that is defined by the experts has a good precision because it is unlikely that wrong instances were placed in any class/relation lists. However, the lexicon defined by the experts can be limited on several aspects:

- It can have very low coverage, thus miss important instances
- It is not a sample of the domain thus it can over-represent certain types of objects and under-represent others

Precision and recall are often combined in a single score which is the harmonic average of *P* and *R*, called **F-score**:

$$F = (1 + \beta^2)\frac{P \cdot R}{R + \beta^2 P}$$

Population and knowledge discovery

NLP can be used for assisting experts in populating the objects defined by the ontology. Machine learning methods for supervised and unsupervised classification can be applied to corpus data to retrieve unknown instances of ontology objects. So far most of the work in this area has concentrated on the problem of finding new members of of a class of objects (cf. Riloff, 1996; Roark & Charniak, 1998), and on finding examples of structural relations such as is-a (Hearst, 1992,Pantel & Ravichandran, 2004) or part-of (Berland & Charniak, 1999). Recent work however has focused also on discovering class attributes (Almuhareb & Poesio, 2004) and arbitrary relation between classes (Ciaramita et al., 2005). Automatic or semi-automatic population of ontology objects is valuable also in terms of evaluation. In fact, it is possible that new senses of already known instances are

discovered, for example because the instance is polysemous/ambiguous (e.g., "Washington" is a person and a location).

Task-oriented evaluations

Ontologies are developed to play a role in information and knowledge management tasks. The most reliable evaluation of an ontology is the quantification of its positive impact on the task performance. In language-related tasks, where ontologies can provide a crucial support for inference, it has been observed that ontology can improve a system's performance; e.g., in Information Retrieval (Welty et al., 2003) and in automatic Question/Answering (Pasca & Harabagiu, 2001). For example, a taxonomical structure of nominal concepts can help a system to find the right answer for a question such as "What flowers did Van Gogh paint?". Hence, a task-oriented evaluation provides first of all empirical means for comparing the performance of a system with and without ontological support. Secondly, it provides a straightforward way of comparing competing ontologies for the same task by comparing their performance.

Furthermore, empirical appllications to high-level tasks such as question answering or information retrieval allow ontology developers to investigate error patterns of the ontology and therefore offer a way of understanding the behavior and limitations of current ontologies in supporting inference. Therefore task-oriented evaluations provide empirical support for further development and improvements of the ontology itself.

2.3.4 Modularity assessment

Another method type is based on the availability of data about the *design* of an ontology. Therefore, it deals with measuring an ontology according to its fitness to an existing repository of reusable components. This makes the measurement very reliable both at design-time, and at reuse-time. On the other hand, modularity can only be assessed easily on ontologies that have been designed with an appropriate methodology.

Modularity assessment requires a specification of reusable components, for example, it requires (one or more) libraries of ontologies, with indications of their provenance, specificity, application history, etc. (cf. also the usability-profile section, 2.4).

Modular designs are not independent: they require that a task and topic assessment has been performed in advance, at least at some level of generality in the case of reusable *generic* components.

Modularity depends on topic assessment, because we need to know what theories are needed in a certain ontology project [Fernandez-Lopez et al. 2004].

Modularity also depends on task assessment, because we need to know how much of a reusable theory is needed. This dependence causes a form of circularity: a reusable component has to be assessed against a task, but it is supposed to provide a ready-made solution to task assessment. For example, if we need a theory of *calendar relations*, we assume that the one we are going to reuse has a built-in task-oriented accuracy, seemingly quite generic. On the other hand, our ontology project' task might require only a fragment of that calendar theory, thus dictating its own task over the reusable component.

There is no trivial solution to this circularity, and a good practice is to isolate the fragment as much as possible, and to import it. This approach is applied more effectively if a reusable component spots its content design patterns, if any [Gangemi 2005].

Stratification

A notable example of modularization architectures employs *reference* and *core ontologies*, which allow to factorize ontology projects, as well as their quality assessment.

For ontologies designed with a *stratified* modularization methodology (cf. Fig. 12), assessment is straightforward, provided that topic and task are clear enough. The typical architecture of stratification requires a foundational layer on which a core ontology is built or reused in accordance to the topic and task of the ontology projects, and domain ontologies are plugged into the core layer.
For ontologies that have no (or a non-stratified) modular design, assessment is much more complex, and requires firstly a modularization procedure, and secondly an assessment with respect to their task and topic. This is typically needed in reengineering projects, e.g. with legacy thesauri or terminologies.
For example, in the Fishery Ontology Service project [Gangemi, Keizer, et al. 2004], legacy fishery thesauri have been reengineered and modularized according to an informal legacy topic hierarchy and core ontology of fishery (Fig. 13). The resulting modules have been matched to existing components for the different topics and tasks addressed by the project. It resulted that, on one hand, the legacy ontologies had a defective modularization, which has been restored by mapping the legacy modules to a newly added core ontology; and, on the other hand, that existing reusable components were not able to provide a satisfactory matching to the very different modules arising from the modularization procedure.

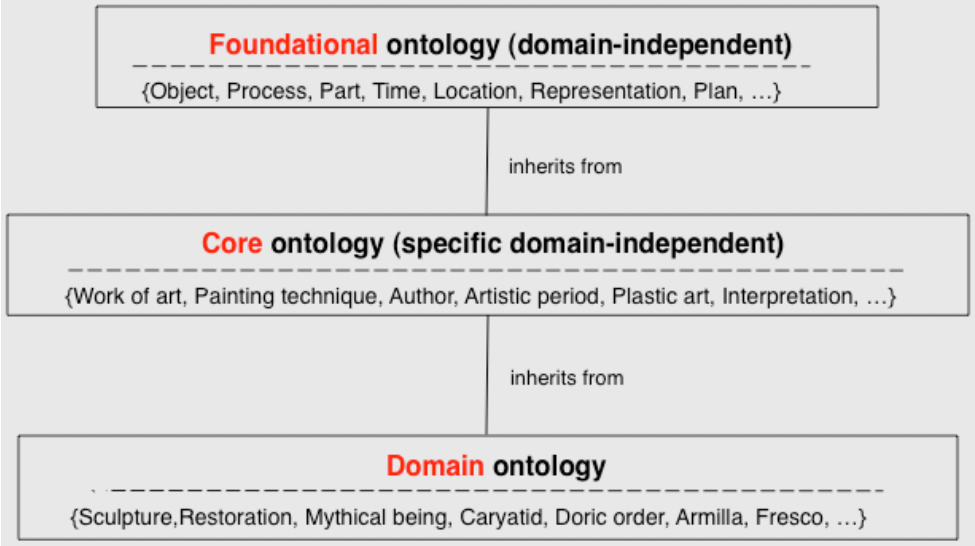

Figure 12. In an ontology for historical works of art, domain ontologies define a vocabulary that is typically stratified in foundational, core, *and* domain *layers*

The results of this evaluation can be summarized by saying that in realistic projects, stratification is usually less 'pristine'. Domain modules often contain bits and pieces of other domains (e.g. geology and law in fishery), thus requiring the reuse of general purpose ontologies like OntoWordNet, which are usually unsatisfactory with reference to agreement and task assessment.

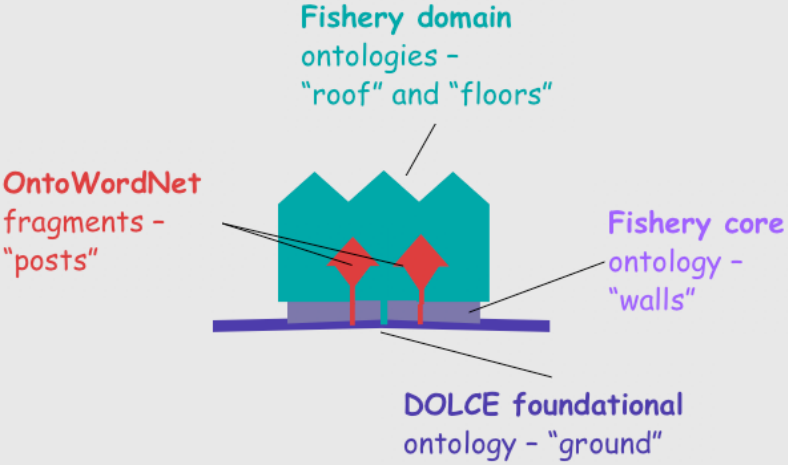

Figure 13. The stratification resulting after modularity checking in the fishery ontology service project. Modules uncovered by existing reusable ontology components have been covered by using a reengineered version of WordNet (a sub-optimal solution).

A final remark should be made on modular design based on *organizational design*, which is discussed in 2.4. This kind of modularity depends on the organization that employs the ontology rather than on off-the-shelf components.

2.4 Measuring the usability-profile of ontologies

Usability-profiling measures focus on the ontology profile, which typically addresses the communication context of an ontology (i.e. its pragmatics). An ontology profile is a set of ontology annotations, i.e. the metadata about an ontology and its elements.
Annotations contain information about structural, functional, or user-oriented properties of an ontology. The structural and functional properties have been presented in 2.2 and 2.3. Some purely user-oriented properties (e.g. *authorship, price, versioning, organizational deployment, interfacing*, etc.) are introduced in this section when needed.
Here we propose some analytical levels concerning usability profiling. The idea is to identify specific parameters to better understand the relations between users and ontologies. The identified analytical levels are **recognition, efficiency,** and **interfacing**.
The added value of this analysis of usability levels is to explicitly match user needs and the ontology development process. Following these usability dimensions, it should be possible to involve a wider variety of users in the effective exploitation of ontologies. Moreover, usability levels are also useful to design specific ontologies for particular user communities.

Recognition annotations

The recognition level makes objects, actions, and options visible. Users need an easy access to the instructions for using ontology in an effective way, and an efficient process to retrieve appropriate meta-information. That is, give your users the information that they need and allow them to pick what they want. Hence recognition is about having a complete documentation and to be sure to guarantee an effective access.
Another point is to not force users to recall information, unless absolutely necessary. Search engines have largely been successful because they shift a memory burden away from users. They store and recall information for users and then, when the results are displayed, users simply perform a recognition task. Since recognition is better than recall, and since overall performance improves, the usability is augmented.

What was the name of your fifth-grade teacher? A question such as this is a request to "recall" information from your long-term memory. If, in addition to this request, you are given a few hints or cues as to the desired name, the memory test becomes a "cued recall" test. If you are given a list of four names, one of which is the name of your teacher, the memory test becomes a "recognition" test (cf. [Kaakinen et al. 2002]).
Ontologies are more usable if they can be recognized appropriately. But information about ontologies cannot be always got from their structure, and functionality tests cannot be replicated so easily. Therefore, ontology recognition requires an adequate set of annotations, which can be preliminarily classified as follows:

1. Annotations (of the overall ontology) about the ontology structure
    - Graph measures
    - Logic-type and computational complexity
    - Meta-consistency
    - Modularization (e.g. owl:imports)
2. Annotations about the ontology function (either at design-time or reuse-time)
    - Lexical annotation of ontology elements (incl. multilingual)
    - Glosses (e.g. rdfs:comment) about ontology elements
    - Agreement status
    - User satisfaction (e.g. http://smi.protege.stanford.edu:8080/KnowledgeZone/) and trust rating
    - Task/use case of the overall ontology (both originally and during its lifecycle)
    - Topic (e.g. rdf:about) of the overall ontology
    - Modularization design of the overall ontology
3. Annotations about the ontology lifecycle (either of the overall ontology, or of its elements)
    - Provenance
    - Methods employed
    - Versioning (e.g. owl:versionInfo)
    - Compatibility (e.g. owl:incompatibleWith)

The annotations from types 1. and 2. concern the structural and functional measures presented in previous sections. The annotations from type 3. are proper *pragmatic* informations (Fig.2), which *profile* the actual usage context of an ontology.
Amount, *completeness*, and *reliability* of annotations are usability measures ranging on the above annotations.
Notice that annotations can be *resident* in an ontology file, *linked* through a URI, *dynamically produced* when needed (e.g. by a local software component, or through a web service), or *retrieved* from an incrementally growing repository (e.g. from a portal that collects users' feedback). Here we abstract out of these different availability systems.

Efficiency annotations

An interesting analytical level concerns the variety of ontology potential uses. Ontology should be designed in order to satisfy the needs of both experienced and inexperienced users. Moreover, it is useful to better understand those possible user behaviours that could drift the ontology use-pattern. Users should be able to achieve their goals in an efficient manner. In order to look at the user's productivity, an analogy to *operating a microwave* is used.

The microwave analogy. People cost a lot more money than machines, and while it might appear that increasing machine productivity must result in increasing human productivity, the opposite is often true. In judging the efficiency of ontology, we need to look beyond just the efficiency of the machine. For example, which of the following takes less time? Heating water in a microwave for one minute and ten seconds or heating it for one minute and eleven seconds? From the standpoint of the microwave, one minute and ten seconds is the obviously correct answer. From the standpoint of the user of the microwave, one minute and eleven seconds is faster. Why? Because in the first case, the user must press the one key twice, then visually locate the zero key, move the finger into place over it, and press it once. In the second case, the user just presses the same key–the one key–three times. It typically takes more than one second to acquire the zero key. Hence, the water is "processed" faster when it is "heated" longer. Other factors beyond speed make the 111 solution more efficient. For example, seeking out a different key not only takes time, it requires a fairly high level of cognitive processing. While the processing is underway, the main task the user was involved with–cooking their meal–must be set aside. The longer it is set aside, the longer it will take to reacquire it.
The managing-operating-balance principle. The principle learnt from the analogy is such that, since typically the highest expense in a business is labor cost, any time the user must wait for the ontology to respond before they can proceed, money is being lost. The balance between *managing instructions* and *operating heating* translates in ontology engineering to a balance between *managing ontologies* at an organizational level and *operating ontologies*.
As a matter of fact, in order to maximize the efficiency of a business or an organization, we need to maximize everyone's efficiency, not just the efficiency of a single group, therefore synchronization and management of distributed ontologies used by different groups is as much important as their local, individual deployment.
Large organizations tend to be compartmentalized, with each group looking out for its own interests, sometimes to the detriment of the organization as a whole. Information resource departments often fall into the trap of creating or adopting ontologies that result in increased efficiency and lowered costs for the information resources department, but only at the cost of lowered productivity for the company as a whole.
The organizational fitness principle. The managing-operating-balance principle boils down to some requisites (*parameters*) for the organization-oriented design of ontology libraries (or of distributed ontologies), which provide constraints to one or more of the following entities: *organization architecture, (complex) application middleware, trading properties, cost, accessibility*, *development effort*. These parameters are defined by the principle of organizational fitness, and are annotated as follows:

1. Annotations (either on the overall ontology, or on ontology elements) about the *organizational design* of a modularized ontology, and about the middleware that allows its deployment
2. Annotations about the *commercial* (trading, pricing) and *legal* (policy, disclaimer) *semantics*
3. Annotations about the *application history* -with reference to development effort (task- or topic-specificity applied to a token scenario) of an ontology

These annotations can also be considered within the *pragmatic* dimension of ontology engineering.
Annotations 2. and 3. (but not 1.) can be applied to isolated ontologies.
Presence, *completeness*, and *reliability* are usability measures ranging on the above

annotations.

Interfacing annotations

The interfacing level concerns the process of matching an ontology to a user interface. As far as evaluation is concerned, we are only interested in the case when an ontology includes annotations to interfacing operations. For example, a *contract negotiation* ontology might contain annotations to allow an implementation of e.g. a *visual contract modelling language*. If such annotations exist, it is indeed an advantage for ontologies that are tightly bound to a certain (computational) service. On the other hand, such annotations may result unnecessary in those cases where an interface language exists that maps to the core elements of a core ontology e.g. for contract negotiation.
Presence, completeness, and *reliability* of interface annotations are further usability measures.

2.5 Ontology validation: qoods, principles, parameters, and trade-offs

Evaluation is important, but within a given project, we may want to *validate* an ontology according to the criteria that are relevant for that project. In practice, we may want to define quality parameters that range over some of the attributes obtained from structural, functional, or purely user-oriented measurement.
In section 2.1 we have already introduced the distinction between *qoods* (quality-oriented ontology descriptions), *principles* (elementary qoods), *value spaces*, *parameters*, dependencies and preference functions between parameters, and provided an example of a *trade-off*, needed when composing principles with conflicting parameters.
In this section, we give a still initial, but more detailed presentation of principles, some of their typical parameters, and an analytic case for a trade-off.

Some principles and parameters

Principles are defined here as structured descriptions of the quality of an ontology ("qoods"): they are considered *elementary* qoods because they usually define a limited set of parameters constraining ontology properties in order to support a common *goal*. Principles also *lack conflicting parameters*.
Here is a list of some principles emerged in the practice of ontology engineering:

- Cognitive ergonomics
- Transparency (explicitness of organizing principles)
- Computational integrity and efficiency
- Meta-level integrity
- Flexibility (context-boundedness)
- Compliance to expertise
- Compliance to procedures for extension, integration, adaptation, etc.
- Generic accessibility (computational as well as commercial)
- Organizational fitness

The parameters defined by principles can be complex, but at the current state of research, they are usually simple scalars ranging on the measurement value spaces provided in 2.2-2-4.

Here is a list of parameters defined by the principles introduced above: for comprehensibility, each parameter is presented with the name of measure on which it ranges, preceded by a + or – sign to indicate the scalar region constrained within the value space:

Cognitive ergonomics. Intuition: this principle prospects an ontology that can be easily understood, manipulated, and exploited. Parameters:
-depth
-breadth
-tangledness
+class/property ratio
+annotations (esp. lexical, glosses, topic)
-anonymous classes
+interfacing
+patterns (dense areas)

Transparency. Intuition: this principle prospects an ontology that can be analyzed in detail, with a rich formalization of conceptual choices and motivations. Parameters:
+modularity
+axiom/class ratio
+patterns
+specific differences
+partitioning
+accuracy
+complexity
+anonymous classes
+modularity design

Computational integrity and efficiency. Intuition: this principle prospects an ontology that can be successfully/easily processed by a reasoner (inference engine, classifier, etc.). Parameters:
+logical consistency
+disjointness ratio
-tangledness
-restrictions
-cycles

Meta-level integrity: Intuition: this principle prospects an ontology that respects certain ordering criteria that are assumed as quality indicators. Parameters:
+meta-level consistency
-tangledness

Flexibility Intuition: this principle prospects an ontology that can be easily adapted to multiple views. Parameters:
+modularity
+partitioning
+context-boundedness

Compliance to expertise Intuition: this principle prospects an ontology that is compliant to one or more users. Parameters:

+precision
+recall
+accuracy

Compliance to procedures for mapping, extension, integration, adaptation Intuition: this principle prospects an ontology that can be easily understood and manipulated for reuse and adaptation. Parameters:
+accuracy(?)
+recognition annotations (esp. lexical)
+modularity
-tangledness(?)

Organizational fitness Intuition: this principle prospects an ontology that can be easily deployed within an organization, and that has a good coverage for that context. Parameters:
+recall
+organizational design annotations
+commercial/legal annotations
+user satisfaction
+organizational design annotations

Generic accessibility Intuition: this principle prospects an ontology that can be easily accessed for effective application. Parameters:
+accuracy (based on task and use cases)
+annotations (esp. policy semantics, application history)
+modularity
-logical complexity

Preference and trade-offs

Due to partly mutual independence of principles, the need for a preferential ordering of quality parameters required by different principles often arises (e.g. because of a conflict, or because two parameters from different principles are unsustainable with existing resources), and sometimes that ordering is actually a trade-off. Trade-offs are needed when two or more principles should be composed. OntoMetric [Lozano-Tello et al. 2004] is an example of a tool that supports measurement based on a preferential ordering.
A trade-off is based on meta-parameters, e.g.: *available resources*, *available expertise*, *business relations*, *tools*, etc.

An example in legal ontologies

We explain with a simple example how trade-offs appear from principle composition.
Transparency and *compliance to expertise* principles usually require *content ontology design patterns* (cf. [Gangemi 2005]), involving *hub nodes* (classes with several properties, cf. [Noy 2004]), then those principles require a *high rate of dense areas* parameter. But dense areas often need the definition of sets of (usually existential) axioms that potentially induce complex (in)direct cycles. Consequently, *high rate of dense areas* depends on a *high complexity* parameter.
The content design pattern for the *LimitViolation* pattern is an example of such a case (Fig.14).

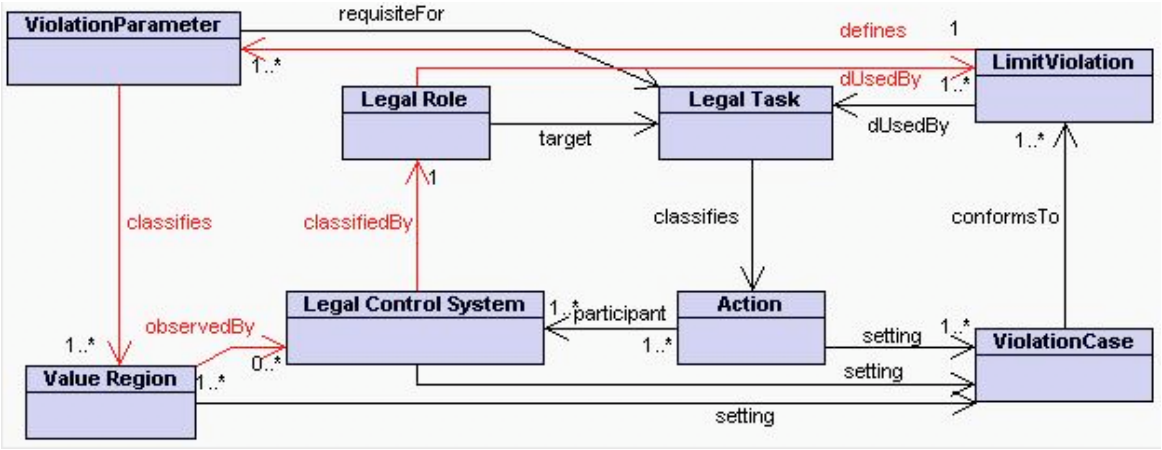

Figure 14. The LimitViolation *pattern in UML, showing a potential indirect cycle: a description of limit violation defines violation parameters ranging on some value space (e.g.,* speed*), also assigning (legal) roles and tasks to legally-relevant entities: control systems, vehicles, persons, actions, etc. A violation case conforms to the description if legally-relevant entities and values are classified by parameters, roles, and tasks.*

The *LimitViolation* pattern contains the following axioms (restrictions) that constitute a cyclical path, encoded here in OWL abstract syntax (corresponding to the red path in Fig.14):

Class(LimitViolation partial restriction(defines someValuesFrom(ViolationParameter)))
Class(ViolationParameter partial restriction(classifies someValuesFrom(ValueRegion)))
Class(ValueRegion partial restriction(observedBy allValuesFrom(LegalControlSystem)))
Class(LegalControlSystem partial restriction(classifiedBy someValuesFrom(LegalRole)))
Class(LegalRole partial restriction(d-used-by someValuesFrom(LimitViolation)))
Class(LimitViolation partial restriction(defines someValuesFrom(ViolationParameter)))
Class(ViolationParameter partial restriction(classifies someValuesFrom(ValueRegion)))
Class(ValueRegion partial restriction(observedBy allValuesFrom(LegalControlSystem)))
Class(LegalControlSystem partial restriction(classifiedBy allValuesFrom(LegalRole)))
Class(LegalRole partial restriction(d-used-by someValuesFrom(LimitViolation)))

If an ontology project using the limit violation axioms is based on a *qood* that aims at both a *transparency* principle, and a *computational efficiency* principle, and we already know (2.1) that it requires a *low rate of cycles parameter* (cf. [Berardi et al. 2001] for the complexity of description logic ports of UML models), then we get a conflict of parameters (Fig.15).
Therefore, a trade-off may be needed in an ontology project that uses the limit violation axioms. The trade-off can be applied by following two approaches.
The first approach defines a *preference ordering* over the parameters, as shown in 2.1, which in the example leads either to accept the complexity, or to dismiss the pattern.
The pattern is in this case essential to the ontology, then, if the *low rate of cycles* is also required because of e.g. available computational resources, we must resort to the second approach: *relaxation of parameters*.
The possible methods to relax the parameters should act on either the reasoning algorithm, or the axioms. Since the first cannot be changed easily in most ontology projects, the best practice is to modify the model according to some *tuning practices* e.g. involving generalization over restrictions, which in our example can be done on one of the following axioms by substituting the class in the restriction with its superclass:

Class(ValueRegion partial restriction(observedBy allValuesFrom(ControlSystem)))
Class(LegalControlSystem partial restriction(classifiedBy allValuesFrom(Role)))

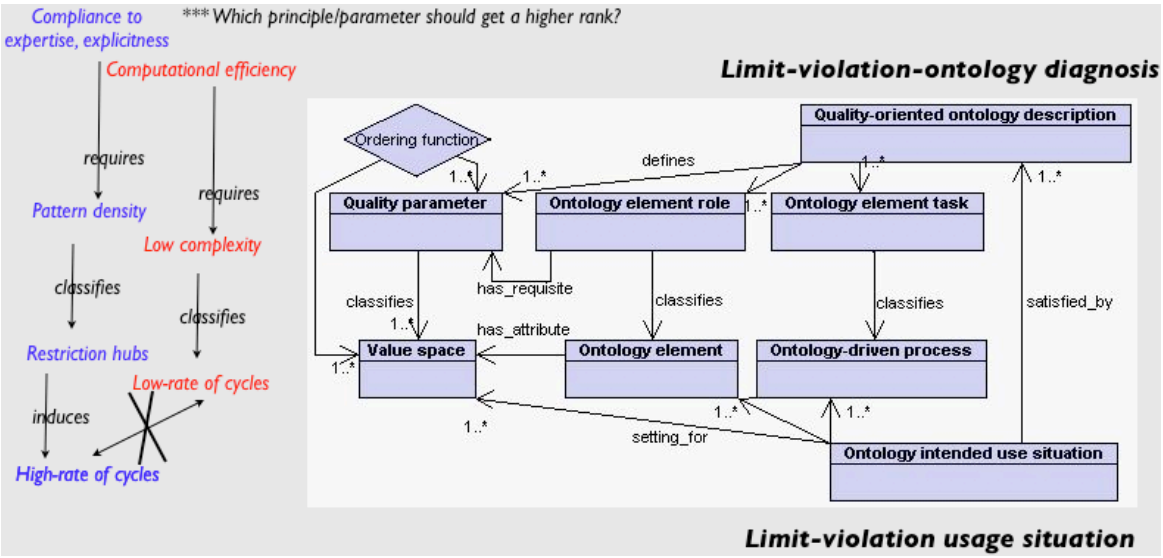

Figure 15. A qood *(a diagnosis of an ontology project using the limit violation pattern) that composes two principles requiring conflicting parameters.*

3. Related work: state of the art in ontology evaluation

As opposed to the apparent simplicity of the four questions we have used to frame the problem of ontology evaluation (see Section 2.1), the available literature on the subject is more complex, fragmentary. Any given approach may address more or less specific versions of our questions, and often more than one of the dimensions we have treated is discussed at the same time.

[Hartmann 2004] tries to systematically disentangle issues by providing a classification-grid for ontology evaluation methods. Such grid is just as general as, though slightly less theoretical than our four questions. It allows to present ontology evaluation methods in terms of answers to the following questions:

- What is the considered method/tool like? Subordinately: what is its goal (Goal)? What functions are supported by it (Function)? At which stage of development of an ontology may it be applied (Application)?
- How useful is the method? Subordinately: for which type of users is it conceived (Users types: Knowledge Engineers, Project Managers, Application Users, Ontology Developers)? How relevant is it to practice (Usefulness)? How usable is it (Usability)? For which type of uses was it conceived in the first place (Use cases)?

In the rest of this section, we review various approaches that are relevant to our work. In order to allow the comparison between different approaches, the two above groups of questions are used as background structure of our review.

3.1 Evaluation by structure measuring

There exists a number of mathematical theories of how to describe and measure (graphical) structures. The most general ones are Graph Theory and Metric Theory. These define notions that are certainly relevant to the problem of ontology evaluation, but their level of abstraction makes them unsuitable for direct application. As opposed to this situation, [Yao et al. 2005] defines a number of Cohesion Metrics that are specific to ontologies.

Cohesion Metrics

According to [Yao et al. 2005] cohesion traditionally refers to the degree to which the elements in a module belong together. In object-oriented software, cohesion refers to the degree of the relatedness or consistency in functionality of the members in a class; strong cohesion is recognized as a desirable property of object-oriented classes because it measures separation of responsibilities, independence of components and control of complexity. Research from software cohesion metrics shows that actually the most cited software cohesion metrics are theoretically based on concepts similar to those of objects. Because cohesion metrics are intended to measure modularity, metrics similar to the software cohesion metrics can be defined to measure relatedness of elements in ontologies.

The authors propose to see ontology cohesion metrics as part of a measure for ontology modularity: ontology cohesion refers to the degree of the relatedness of OWL classes, which are semantically/conceptually related by the properties. An ontology has a high cohesion value if its entities are strongly related. The idea behind this is that the concepts grouped in an ontology should be conceptually related for a particular domain or a sub-domain in order to achieve common goals.

In terms of the two groups of questions indicated in the introduction to this section, the *goal* of Cohesion Metrics is to measure relatedness of elements in OWL ontologies. The *definitions* that support the proposed metrics are the following:

- C_1, C_2, …, Cm be the set of m classes explicitly defined in an ontology.
- P_1, P_2, …, Pn be the set of n properties which work as relationship between the set of classes.
- F_{c1}, F_{c2}, …, F_{cm}, be the fanout of each class C_i in the set.
- O_i be an OWL ontology of interest.
- → be subtype relationship from C_i to C_j such that $C_i \rightarrow C_j$ if class C_j is a subclass of class C_i .
- A be a finite set and T be a relation, then T is a tree if there is a vertex V_0 in A and there exists a unique path in T from V_0 to every other vertex in A.
- A be a finite set and T be a relation on A. A vertex V_n has a Fanout of degree m if there exist m relationships to other vertices in A.
- A be a finite set and T be a relation on A. A vertex V_q is called a leaf of the tree if it has Fanout of degree 0.

Three are the Coehsion Metrics, i.e. the *functions*, defined in terms of the notions given above:

Number of Root Classes (NoR) is the number of root classes explicitly defined in the ontology O_i . A root class in an ontology means the class has no semantic super class explicitly defined in the ontology O_i. Mathematically, NoR can be formulated as follows:

$NoR(O_i) = \Sigma\ C_j$ for all $1 \leq j \leq n$ (number of root classes in O_i)

Number of Leaf Classes (NoL) is the number of leaf classes explicitly defined in the ontology O_i. A leaf class in an ontology means the class has no semantic subclass explicitly defined in the ontology O_i. Mathematically, NoL can be formulated as follows:

$NoL(O_i) = \Sigma\ L_j$ for all $1 \leq j \leq n$ (number of leaf classes in O_i)

Average Depth of Inheritance Tree of Leaf Nodes (ADIT-LN) is the sum of depths of all paths divided by the total number of paths. A depth is the total number of nodes starting from the root node to the leaf node in a path. The total number of paths in an ontology is all distinct paths from each root node to each leaf node if there exists an inheritance path from the root node to the leaf node. And root node is the first level in each path. For example, ADIT-LN of an OWL ontology is described in Fig. 3. Mathematically, ADIT-LN is formulated as follows:

$ADIT\text{-}LN(O_i) = \Sigma\ D_j / n$ for all D_j (D_j is total number of nodes on jth path)
for $1 \leq j \leq n$ (number of paths in O_i)

3.2 Evaluation by function measuring

Most of the literature on ontology evaluation focuses on functionality-related issues. The functionality of an ontology is mostly measured by evaluating its appropriateness as semantic backbone of either decision-support or information systems that operate in the domain represented by the ontology.

In the following, we consider some analyses and proposed methods for evaluating the functionality of an ontology: OntoMetric, OntoClean, EvaLexon, Methontology, Content Evaluation.

OntoMetric

OntoMetric [Lozano-Tello et al. 2004] is an adaptation of the Analytic Hierarchy Process, i.e. a mathematical method for scaling priorities in hierarchical structures. The main *goal* of this method is to help choose the appropriate ontology for a new project. The *functions* supported by OntoMetric are the ordering by importance of project objectives, the qualitative analysis of candidate ontologies for the project, the quantitative measure of the suitability of each candidate. The *application* of OntoMetric can only follow ontology release. The method is meant for *users types* like Engineers or Project Managers who need to look for ontologies over the Web at the purpose of incorporating them into their systems. Therefore, OntoMetric makes itself *useful* as a support to the evaluation of the relative advantages and risks of choosing an ontology over others.
The main drawback of OntoMetric is related to its *usability*: specifying the characteristics of an ontology is complicated and takes time; assessing its characteristics is quite subjective. On top of this, the number of *use cases* is limited, which is an important obstacle to defining (inter-subjective or objective) parameters based on a large enough number of comparable cases.

OntoClean

As opposed to OntoMetric, OntoClean [Welty et al. 2001] is meant for application at the pre-modelling and modelling stages, i.e. during ontology development. The main *goal* is to detect both formal and semantic inconsistencies in the properties defined by an ontology. The main *function* of OntoClean is the formal evaluation of the properties defined in the ontology by means of a predefined ideal taxonomical structure of meta-properties.

EvaLexon

Similarly to OntoClean, EvaLexon finds *application* at the pre-modelling/modeling stage [Spyns 2005]. The main *goal* here is to evaluate at development time ontologies that are created by human beings from text. In sharp contrast with OntoClean, EvaLexon is meant for linguistic rather than conceptual evaluation. Its main *function* is the measurement of how appropriate are the terms (to be) used in an ontology. A term is judged more or less appropriate depending on its frequency both in the text from which the ontology is (being) derived and in a list of relevant domain-specific terms. Regression allows for direct and indirect measurement of the ontology's recall, precision, coverage and accuracy.

Task-based approach

In [Porzel et al. 2004] a linguistics-based approach partly comparable to EvaLexon is defined.
The *goal* of the proposal is to evaluate ontologies with respect to three basic levels: vocabulary, taxonomy and (non-taxonomic) semantic relations. As these levels are also subject to different respective learning approaches - the common notion of error rates, such as found in word - or concept-error rates suffices for each level of evaluation. The resulting task-based evaluation should show either of the following shortcomings:

- insertion errors indicating superfluous concepts, isa- and semantic relations;
- deletion errors indicating missing concepts, isa- and semantic relations;

- substitution errors indicating off-target or ambiguous concepts, isa- and semantic relations.

Moreover, given appropriate tasks and maximally independent algorithms operating on the ontology in solving these tasks and given the task evaluation gold standards, the error rates can be calculated that correspond to specific ontological shortcomings of the translation of error rates to the three basic ontological levels.By applying this evaluation scheme, improvements in the ontology can be tested and measured that are brought about by learning approaches that target the same levels and issues in the ontology learning and population field.
Similarly to EvaLexon, the *functions* proposed here are based on two key arguments: the task and the gold standard. The task needs to be sufficiently complex to constitute a suitable benchmark for examining a given ontology. Especially if the target of the evaluation is to include non-taxonomic relations as well, tasks are needed where the performance outcome hinges substantially on the way these relations are modeled within the ontology. The gold standard is a perfectly annotated corpus of part-of-speech tags, word senses, tag ontological relations, given sets of answers (so-called keys) used to evaluate the performance of algorithms that are run on the ontology to perform the task.

Methontology

The intended *users* of the Methontology framework [Fernández-López et al 2004] are domain experts and ontology makers who are not familiar with implementation environments. The *goal* is to let them build ontologies from scratch. To this end a number of *functions* are provided that enable easier intermediate representations of ontologies. Such representations are meant to bridge the gap between how people think about a domain and the languages usually used to define ontologies at the formal level. In other words, Methontology makes it possible to work on ontologies at the knowledge level only, and it does so by supporting functions like the specification of the ontology development process as well as of its life-cycle (based on evolving prototypes); the specification of ontologies at the knowledge level; the multilingual translation that automatically transforms the specification into several target codes.
Methontology is well exemplified by the following specification of a process of ontology re-use:

- *specifying the requirements* the ontology must satisfy in the new application (purpose, language in which it is needed, key aspects that should be modelled, scope – the latter defined through competency questions);
- *searching an ontology* that covers most of the identified necessities;
- *adapting the chosen ontology* so that it satisfies the necessities completely;
- *integrating the ontology* in the system (this may involve language translation).

A concrete example of how the intermediate representations work is provided in [Blazquez et al. 1998]. Here the intermediate levels are used to facilitate the selection of time ontologies by non-experts. On the one hand, the article shows how such levels consist of very intuitive and basic definitions of entities and relations, which emerge from main(stream) theories of time and are visualized through tables and graphs. On the other hand, the article points at the scarcity of ready-for-use intermediate conceptual models of time ontologies, a serious obstacle for the analytical work required by the specification process spelled out above.

Content and ontology technology evaluation

In the context of a discussion on functional and usability-related aspects of ontology evaluation [Gómez-Pérez 2003] a distinction is drawn between two main evaluation dimensions: content evaluation and ontology technology evaluation. This distinction may well be used for classifying the ontology evaluation methods introduce so far, as well as for framing the issue of ontology evaluation in general terms.

On the one hand, *content evaluation* is related to the KR paradigm that underlies the language in which the ontology is implemented (be it RDF schemas, description logic, first order logic, etc.). As already pointed out for methods like OntoClean or Methontology, the *goal* of content evaluation is to detect inconsistencies or redundancies before these spread out in applications. The *application* of content evaluation techniques should take place during the entire ontology life-cycle, as well as during the entire ontology-building process. *Functions* should support the evaluation of concept taxonomies, properties, relations and axioms

On the other hand, *ontology technology*, i.e. ontology development tools like OILed and Protégé, should be subject to evaluation too. Here the *goal* is to ensure smooth and correct integration with industrial software environments. The *application* of such evaluation should be directed at the expressiveness of the KR model underlying the ontology editor; the tool's interoperability, in terms of quality of import/export functions (i.e. how much knowledge is lost with format transformation), scalability (i.e. how different building platforms scale when managing large ontologies with thousands of components, as well as time required to open and save, etc.), navigability (e.g. how easy it is to search for a component), usability (e.g. user interfaces' clarity and consistency), and available content evaluation functions. There are no implemented *functions* for evaluation ontology technology, but systematic comparisons between tools have been conducted by SIG3 (Special Interest Group on Enterprise Standard Ontology Environments).

Based on the distinction between content and ontology technology evaluation, a number of general conclusions may be drawn:

- content evaluation needs specific methods when applied to ontology components other than taxonomies;
- experimental results in ontology technology evaluation show that tools based on similar knowledge models are more interoperable because they preserve more knowledge throughout the knowledge exchange process;
- the most well-known ontology development tools (OILed, OntoEdit, Protègé, etc.) support content evaluation mainly in the form of circularities detection. This though is not enough for more-than-trivial content evaluation;
- language-dependent evaluation tools should be developed, in order to be used by different ontology platforms, and make these able to detect errors in an ontology written in the traditional or Semantic Web languages before it is imported.

On content evaluation, [Daelemans et al 2004] points out how recently developed NPL techniques can be – and currently are – used for evaluating ontologies' semantics (vs their syntax). NLP not only helps content collection from huge amounts of text and maintenance, but it also provides the means for showing that ontologies indeed represent "consensual conceptualizations and not just one person's ideas".

In particular, information extraction, named entity recognition and shallow parsing, combined with NLP's standard pattern-matching and machine-learning techiniques, allow for semi-automatic extraction of ontological knowledge (concepts and relations) from texts. For instance, in the context of the Ontobasis project on the domain of the Medline abstract language, knowledge is extracted by using a (completely automatic) shallow

parser and by then applying clustering techniques for grouping semantic similarities and dependencies into classes. The pivot idea behind these techniques is that all "terms with similar syntactic relations to other terms are semantically related", which allows to *semi-automatically* make explicit meanings that are implicit in the syntactic relations of a term. Classes that are generated by this method are then used as a basis for the extraction of new relations through pattern-matching rules. However, both in the clustering step and the pattern-matching one human intervention is still essential to evaluate the proposed ontological structures. "The difference with purely handcrafted ontology development is that.. [such evaluation].. is much easier, more complete and faster than inventing ontological structures".

3.3 Evaluation by usability measuring

In [Noy 2004] it is argued that, although most structural and functional evaluation methods are necessary, none are helpful to ontology consumers, who need to discover which ontologies exist and, more important, which ones would be suitable for their tasks at hand. Knowing whether an ontology is correct according to some specific formal criteria might help in the ultimate decision to use an ontology but will shed little light on whether or not it is good for a particular purpose or task. What is needed is not only a system for evaluating ontologies objectively from some generic viewpoint, but also practical ways (*function*) for ontology consumers to discover and evaluate ontologies. Information such as the number of concepts or even an ontology's complete formal correctness is probably not the most important criteria in this task (although it is often the easiest to obtain).
Based on this considerations alternative techniques are proposed, as follows:

- *Ontology summarization*: To decide whether to buy a book, we read the blurb on the book jacket; to decide whether a paper is relevant to our work, we read its abstract. To decide whether a particular ontology fits our application's requirements, we need some abstract or summary of what this ontology covers. Such a summary might include a couple of top levels in the ontology's class hierarchy—perhaps a graphical representation of these top-level concepts and links between them. We can generate these top-level snapshots automatically or let ontology authors include them as metadata for an ontology. The summary can also include an ontology's hub concepts—those with the largest number of links in and out of them. What's more interesting, we can experiment with metrics similar to Google's PageRank: the concept is more important if other important concepts link to it. This computation can take into account specific links' semantics (giving a subclass-superclass link a lower value than a property link, for instance) or exclude some links or properties. By experimenting with these measures, we can discover which ones yield the concepts that users deem important. The hub concepts are often much better starting points in exploring and understanding an ontology than the top level of its class hierarchy.
- *Epinions for ontologies*: in addition to reading a book's blurb to determine if we want to buy it, we often read reviews of the book by both book critics and other readers. Similarly, when choosing a movie or a consumer product, such as a coffee maker or a pair of skis, we use the Web to find others' opinions. A network for ontologies would help guide our ontology-consumer friend in finding whether a particular ontology would be suitable for his or her project. The reviews should include not only an ontology's qualitative assessment (Is it well developed? Does it have major holes? Is it correct?) but also, and perhaps more important, experience reports.

- *Views and customization*: To evaluate an ontology properly, users might need to see a view of an ontology that takes into account their expertise, perspectives, the required level of granularity, or a subset of the domain the ontology they're interested in covers. If we can let ontology developers annotate concepts and relations with information about which perspectives these terms and relations should appear in and how to present or name them, we'll be able to present these different perspectives automatically. Similarly, an ontology developer might want to indicate that certain concepts or relations should be displayed only to users who identify themselves as experts (presenting a simpler, trimmed down view for novices). For an ontology consumer, it's often much easier to evaluate a smaller ontology with only the concepts related to his or her concepts of interest than to evaluate a large general reference resource.

4. Conclusions and future work

In this report, we have attempted a comprehensive framework for evaluating and validating ontologies. We have chosen *semiotics* as a perspective appropriate to distinguish the different aspects of ontology engineering in practice.

Structural, functional, and profiling properties of ontologies have been separately addressed, and a design pattern for *quality-oriented ontology descriptions* (*qoods*) has been proposed, which allows to model *construction principles*, to define *quality parameters* on ontology properties, to state parameter *dependencies*, and to compose principles by means of *preference orderings or relaxation* on parameters, in the context of specific ontology projects.

Some areas of the report need refinement and a richer set of examples, and the state of art will be enlarged to cover a larger part of the rapidly growing literature.

Future work will focus on the empirical assessment of the framework, e.g. by measuring existing ontologies, by building an ontology on a same task/topic but by using different qoods, and by creating correlations between user-oriented and structural measures.

Another area of research is the creation of tools to assist in the detection of metrics, user annotations, and qood-deployment within an ontology project. The OntoMetric system [Lozano-Tello et al 2004] is a first component to be reused in order to start such an implementation.

References

Almuhareb A., Poesio M., 2004: “Attribute-based and value-based clustering: an evaluation”. In Proceedings of the Conference on Empirical Methods in Natural Language Processing.

Baeza-Yates R. and Ribeiro-Neto B., 1999: “Modern Information Retrieval”. Addison Wesley.

Berardi D., Calvanese D., De Giacomo G., “Reasoning on UML Class Diagrams using Description Logic Based Systems”. In Proc. of the KI'2001 Workshop on Applications of Description Logics, 2001.

Berland M. and Charniak E., 1999: "Finding parts in very large corpora." In Proceedings of ACL'99.

Blazquez M., Fernandez M., J. Garca-Pinar M., and Gomez-Perez A. *Building Ontologies at the Knowledge Level Using the Ontology Design Environment*. In Proceedings of the Knowledge Acquisition Workshop, KAW98, 1998.

Brewster C., Alani H,, Dasmahapatra S. and Wilks Y.: “Data-driven ontology evaluation”. Proceedings of LREC 2004.

Catenacci C., Ciaramita M. Gil R., Gangemi A., Guarino N., and Lehmann J., “Ontology evaluation: A review of methods and an integrated model for the quality diagnostic task”, Deliverable of the OntoDev Italian Ministry of Research Project, available as TechReport at http://www.loa-cnr.it/Publications.html, 2005.

Ciaramita M., Gangemi A., Ratsch E., Saric J., and Rojas I., 2005: “Unsupervised Learning of Semantic Relations between Concepts of a Molecular Biology Ontology”. In Proceedings of the 19th International Joint Conference on Artificial Intelligence.

Daelemans W., Reinberger M.L. , 2004: “Shallow Text Understanding for Ontology Content Evaluation” . IEEE Intelligent Systems 1541-1672, 2004.

Eco U., 1984: “Semiotica e filosofia del linguaggio”. Einaudi.

Fernández-López M. and Gómez-Pérez A., 2004: “Searching for a Time Ontology for Semantic Web Applications”. In Formal Ontology in Information Systems, A.C. Varzi and L. Vieu (Eds.), IOS Press.

Gangemi, A., F. Fisseha, J. Keizer, J. Lehmann, A. Liang, I. Pettman, M. Sini, M. Taconet, 2004: “A Core Ontology of Fishery and its Use in the FOS Project”, in Gangemi, A., Borgo, S. (eds.): Proceedings of the EKAW*04 Workshop on Core Ontologies in Ontology Engineering. Available from: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS//Vol-118/.

Gangemi A., Borgo S., Catenacci C., and Lehmann J., “Task taxonomies for knowledge content”, Deliverable D07 of the EU FP6 project “Metokis”, available at http://www.loa-cnr.it/Publications.html, 2004.

Gangemi A.: “Ontology Design Patterns for Semantic Web Content”. In Motta E. and Gil Y., Proceedings of the Fourth International Semantic Web Conference, 2005.

Gluck M. and Corter J., 1985: “Information, Uncertainty and the Utility of Categories”. In Proceedings of the 7th Annual Conference of the Cognitive Science Society.

Gómez-Pérez A.: “Ontology Evaluation”, in Handbook on Ontologies, S. Staab and R. Studer, eds., Springer-Verlag, 2003,pp. 251–274.

Guarino N.: “Towards a Formal Evaluation of Ontology Quality”. IEEE Intelligent Systems 1541-1672, 2004.

Guarino N. and Welty C.: “Evaluating Ontological Decisions with OntoClean”, Comm. ACM, vol. 45, 2, 2002, pp. 61–65.

Guarino N. and Giaretta P.: “Ontologies and Knowledge Bases”, Mars N. (ed.) Towards Very Large Knowledge Bases, Amsterdam, IOS (1995).

Hartmann J., Spyns P., Giboin A., Maynard D., Cuel R., Suárez-Figueroa M.C., and Sure Y., “Methods for ontology evaluation”. Knowledge Web Deliverable D1.2.3, v. 0.1 (2004).

Huang Z., van Harmelen F., and ten Teije A., 2005: “Reasoning with Inconsistent Ontologies”, IJCAI05.

Jakobson R.: “Linguistics and Poetics: Closing Statement”. In: Style in Language. MIT Press, Cambridge,

MA (1960)

Kaakinen, J., Hyona, J., & Keenan, J.M. (2002). Individual differences in perspective effects on on-line text processing. *Discourse Processes*, 33, 159 - 173.

Lozano-Tello, A. and Gomez-Perez A., 2004: "ONTOMETRIC: A method to choose the appropriate ontology", J. of Database Management, 15(2).

Maedche A. and Staab S.: "Measuring the similarity between ontologies". Proc. of the 13th Conf. on Knowledge Engineering and Knowledge Management, Springer, Berlin, 2002.

Masolo, C., A. Gangemi, N. Guarino, A. Oltramari and L. Schneider: WonderWeb Deliverable D18: The WonderWeb Library of Foundational Ontologies (2004).

Noy, N., "Defining N-ary Relations on the Semantic Web: Use With Individuals", W3C Working Draft: http://www.w3.org/TR/swbp-n-aryRelations/ (2005).

Noy, N., "Evaluation by Ontology Consumers". IEEE Intelligent Systems 1541-1672, 2004.

Oberle D., Gangemi A., Mika P., and Sabou M., "Foundations for service ontologies: Aligning OWL-S to DOLCE", in Staab S. and Patel-Schneider P. (eds.), Proceedings of the World Wide Web Conference (WWW2004), Semantic Web Track, (2004).

Pantel P. and Ravichandran D., 2004: "Automatically Labeling Semantic Classes". In Proceedings of HLT-NAACL 2004.

Pasca M. and Harabagiu S.H., 2001: "The Informative Role of WordNet in Open-Domain Question Answering". In NAACL 2001 Workshop on WordNet and Other Lexical Resources: Applications, Extensions and Customizations.

Peirce, Charles (1931-1958). *Collected Papers*, vols. 1-8, C. Hartshorne, P. Weiss and A.W. Burks (eds). Cambridge, MA: Harvard University Press.

Porzel R. and Malaka R.: "A Task-based Approach for Ontology Evaluation". Proc. of ECAI 2004.

Riloff E., 1996: "An Empirical Study of Automated Dictionary Construction for Information Extraction in Three Domains". Artificial Intelligence, 85.

Roark B., and Charniak E., 1998: "Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction." In Proceedings of COLING-ACL'98.

Spyns P., EvaLexon: Assessing triples mined from texts. Technical Report 09, STAR Lab, Brussel, 2005.

Steels L.: "Components of Expertise", AI Magazine, 11, 2, 1990, pp. 30-49.

Sugiura N., Shigeta Y., Fukuta N., Izumi N., and Yamaguchi T., 2004: "Towards On-the-fly Ontology Construction Focusing on Ontology Quality Improvement", ESWC04.

Sure Y. (ed.), 2004: "Why Evaluate Ontology Technologies? Because It Works!", IEEE Intelligent Systems 1541-1672.

Uren V., Buckingham Shum S, Mancini C. and Li G.: "Modelling Naturalistic Argumentation in Research Literatures". Proceedings of the 4th Workshop on Computational Models of Natural Argument, 2004.

Uschold U. and Gruninger M.,"Ontologies: Principles, Methods, and Applications," Knowledge Eng. Rev.,vol. 11,no. 2,1996,pp. 93–155.

Vasiliu L., Moran M., Bussler C., Roman D., "WSMO in DIP", DIP EU FP6 Project Deliverable D19.1, v0.1, available at http://www.wsmo.org/2004/d19/d19.1/v0.2/, 2004.

Welty C., Guarino N., "Supporting ontological analysis of taxonomic relationships", *Data and Knowledge Engineering* vol. 39, no. 1, pp. 51-74, 2001] and [N. Guarino and C. Welty, "Evaluating Ontological Decisions with OntoClean," Comm. ACM, vol. 45, no. 2, 2002, pp. 61–65.

Welty C., Kalra R., and Chu-Carroll J., 2003: "Evaluating Ontological Analysis". In Proceedings of the ISWC-03 Workshop on Semantic Integration.

Yao H., Orme A.M., and Etzkorn L., 2005: "Cohesion Metrics for Ontology Design and Application", Journal of Computer Science, 1(1): 107-113, 2005.