Top-Level Ontology of Knowledge Sharing Criteria

Philippe A. MARTIN
ESIROI I.T., EA2525 LIM, Uni. of La Réunion, Sainte Clotilde, France
(+ adjunct researcher of the School of ICT, Griffith Uni., Australia)

Abstract. This article presents a generic model and a top-level ontology to represent, organize and combine knowledge from various sources and, in particular, best practices and quality criteria or measures for knowledge sharing. These elements may then also be used by people to compare tools, techniques and knowledge providers. One top-level distinction permitting to organize this ontology is the one between content, medium and containers of descriptions. Various structural, ontological, syntactical and lexical distinctions are then used.

Keywords: Knowledge Representation/Comparison/Organization/Evaluation

1 Introduction

How should data or knowledge be represented and published so it can be most easily retrieved, re-used and managed? Then, how to compare or evaluate *knowledge* statements, bases, management techniques, management tools and providers? To answer those questions, many knowledge sharing/engineering supporting elements have been proposed: approaches [1-2], languages [3-7], ontologies [8-13], methodologies [14-15], best practices or design patterns [16-18], categories of evaluation criteria or measures [19-22], evaluation queries [23-25], benchmarks [26-27], techniques [28-xx], software [xx], etc. (Note to the reviewers: in this article, "xx" is used to preserve the anonymity constraints). However, it is difficult to compare these elements or their sub-elements, choose between them, combine them or have a *synthetic organized view* of what can or should be done for knowledge sharing purposes. Indeed, these elements often do not use similar terminologies or categorizations and no *ontology or library* has been proposed to compare, index, organize and generalize such elements (which are at various levels of abstraction and may be contradictory). The top-level ontology presented in this article is a step in that direction. This article refers to it as "this knowledge criteria/quality ontology" or simply "this (top-level) ontology". It is focused on elements related to Linked Data [2].

As with any other ontology, the bigger *and more organized* it will become, the more useful it will be for all the previously cited tasks. (For the knowledge operationalization tasks, bigger is no longer better but eases the selection of knowledge for modules of relevant sizes and content for an application; this article and its top-level ontology do not really address this phase, only the knowledge sharing tasks.) This ontology is published on-line via a knowledge server [xx] and can be cooperatively extended by any Web user via this server. This server uses editing protocols and an abstract model allowing its knowledge base (KB) to be consistent and organized even though knowledge statements come from different sources and hence can contradict each other. The use of such a model can be seen as a "best practice" (for knowledge sharing). This abstract model is also

required for organizing the various previously cited elements of the top-level ontology in a scalable way and allow their combination by the users in a flexible way. Hence, this article presents this model and introduces its recently formalization version in Section 2.

Since "better" is often application-dependent and user-dependent, this ontology does not use "better" relations between its elements. However, Section 2 show how "default rules" can ease the definition and aggregations of such relations by users. Relations such as "(has for) more precision" can be exploited by end-users' functions for measuring or comparing the quality of elements. This ontology does not use specialization or "part of" relations to organize knowledge management tools and techniques (e.g., for knowledge extraction, retrieval, matching, merging, representation, inferencing, validation, edition, annotation, modularization and publishing). Indeed, this would be a huge task and, especially at a general level, these processes are so intertwined that they are difficult to distinguish and organize in a *scalable* way, i.e., in a systematic and non-arbitrary way within a specialization hierarchy and a part-of hierarchy. Here, "non-arbitrary" implies the use of conceptual distinctions that are clear enough to lead different persons to categorize a same thing at a same place in a specialization/part-of hierarchy (note: a hierarchy does not have to be a tree); this significantly reduces implicit redundancies [14].

A clear top-level partition of *information objects* is the distinction between description-content, description-medium and description-container objects. *Description-content* objects are conceptual categories, as well as formal/informal terms or statements referring to or defining these categories. They are interpretations or abstractions of a (real or imaginary) situation or object. E.g.: abstract models, ontologies, terminologies, languages and any of their sub-elements (e.g., the concept/relation types of RDF and OWL). D*escription-medium* objects are concrete model objects permitting to visually/orally/... present description-content objects. E.g.: graphical interface objects and syntax/style objects such as those specified by XML, CSS and XSLT. *Description-container* objects are the other information objects, i.e., non-physical objects permitting to store and manage description-content and description-medium objects. E.g.: files, file repositories, distributed databases and file servers. Since the knowledge quality ontology of this article is about information objects, the above partition is at its top. Then, knowledge management tools and techniques can be compared and evaluated with respect to (wrt) the qualities of the information objects that they allow as input and output, or lead their users to produce. Similarly, knowledge providers can be evaluated based on the information objects they have provided. Section 2 illustrates the way quality measures and their combination can be specified incrementally. The sections 3, 4 and 5 are respectively about the evaluation of description-content, description-medium and description-container. These sections relate, organize and generalize knowledge sharing best practices and quality criteria or measures from various sources. Some categories from each of the above referred articles are included in the ontology. [17-22] include the most complete lists of high-level categories that seemed to exist so far for Linked Data. All their categories are integrated in the ontology. The various sources sometimes had categorizations. Section 6 shows those of the four most organized sources that seemed to exist for Linked Data so far. However, these categorizations are essentially only two levels deep and not always intuitive. Additional categorizations would be interesting, especially if they are clear-cut (then, the more categories, the better).

2 A Formal Base for Flexible Knowledge Sharing and Comparison

The model proposed by this article relies on the model of KIF 3.0 [5], a LISP-based relatively intuitive first-order-logic language that became popular for formalizing and translating knowledge representation languages (KRLs) and ontologies. The used KIF syntax is the one of KIF 3.0 plus the new constructs introduced by dpANS KIF [6]. The model and syntax of KIF (Knowledge Interchange Format) were respectively mostly reused by the Common Logic standard and its CLIF notation [7] but without features for specifying meta-statements, definitions and (monotonic or not) inference rules. First-order statements and some 2nd-order-looking ones can be expressed in KIF, and hence all statements that can be expressed with the W3C KRLs (RDF+OWL/XML, N3, RIF, …).

With this model, a KB is a set of statements which are partial/complete definitions, sentences or rules [5]. Statements use terms (called "constants" in KIF; they refer to conceptual categories). Definitions are descriptions which may be said to be "neither true nor false" or "always true by definition". Non-conservative definitions are those which may introduce an inconsistency when added to the KB (like sentences which are "commands" but not information retrieval queries). The model used in this article considers them as shortcuts for a definition plus a sentence. Thus, more precisely, it considers a *statement as either a (conservative) definition, a belief or a command (rule or sentence that is not a belief)*. A belief is a sentence that has a meta-statement indicating its source: authoring agent, interpreter or source KB. If an authoring agent is specified, it is assumed that he believes the embedded statement. A belief may be false (purposely if its authoring agent lie) and hence, unlike a definition, can be *contradicted* by the belief of another source. However, this does not lead to a (logical) *inconsistency* in the KB: a sentence stating that someone believes something is not inconsistent with a sentence stating that someone else believes something opposite. If a same person states opposite beliefs, there is a (detected or not) inconsistency in the KB (if the periods of the beliefs are represented, another condition is that these periods overlap). A meta-statement *contextualizes* a statement if it adds "contextualizing relations" to it, that is, constraints without which the statement *may be* false. Only a belief, not a definition, can be used to express that a particular meaning is the *most popular* meaning of an informal term (i.e., a word with more than one meaning) in a certain community. A query and a KB policy can be represented as definitions. They can also be represented as commands, hence stated in a way that may lead to inconsistencies.

In a shared KB, terms and statements may come from various sources. The KB must be consistent and have a KB consistency/organization policy. Thus, except for queries and statements expressing this policy, a shared KB management system (KBMS) must only accept statements and terms that are "contextualized by their sources". For terms, this is a lexical contextualization, e.g., via their prefixing by an identifier of the source, as with xsd:float (with the XML namespace shortcut) and http://url_identifying_the_source#term_identifier_in_the_source_namespace. For statements, this may be done by using different description containers for different sources. However, the source or other contextualizing information about the contained statements still needs to be represented. The direct use of meta-statements is more flexible. The source of a definition needs not be made explicit if this source is the same as the source of the defined term. Otherwise, it must be made explicit: this is not a definition but actually a belief since this is an assumption of someone about what the source of the term means via this term. A shared KB must then have

a type such as xx:contextualizing_relation, with various subtypes to list and organize such relations, e.g., “belief source” relations and temporal or modal relations on the represented situation. “Statement creation date” relations are also required by cooperation protocols and quality measures but are not contextualizing.

For convenience and security purposes, a KBMS should not ask or accept a user to specify “source users” and creation dates for the beliefs he adds or imports into the KB. If these beliefs are not yet recorded, this user is their “source user” and first believer. A “source document” may also be recorded. The KBMS may impose or encourage users not to enter certain kinds of statements, those that do not satisfy certain quality criteria selected by the KBMS authors or by the users (to have the KBMS help them create more re-usable statements). Creation dates are added by the KBMS when accepting statements.

The above summarized model is formalism independent. However, it has been formalized in KIF (note added on the 23/05/2013: the core of this formalization is at http://www.webkb.org/kb/it/d_KSmodel.html). The following excerpts are i) the top function allowing the KBMS to validate a statement according to the above model, ii) the top relation for handling contextualizations ('%' is the chosen ending for source prefixes within terms; ':' and '#' have other meanings in KIF), iii) the top function for dividing a statement into its “atomic substatements”, i.e., statements that cannot be divided into “contextualized AND-clauses” (indeed, no (sub-)statement should be handled without its contextualization), iv) a way to represent “default rules” and v) a way for the KBMS to express who the current end-user is. Except for operators, all identifiers are nominal expressions (this is a best practice later listed). For a “concept type” (“unary relation” in KIF, “class” in RDF), the initial is in uppercase. For a “relation type” (“function” or “non-unary relation” in KIF, instance of rdf:Property if it is binary), the initial is in lowercase. Terms without prefix are KIF terms.

```
;;During the parsing of a new statement ,knowing its asserter and the date, the KBMS should
;;call the next defined function on this statement to check the statement wrt the KBMS policy
;;and the criteria selected by the (end-)user. The user's statements must be represented via
;;either xx%def_rel, xx%def_fct_rel, xx%def_fct or any subtype of the relation type xx%source.
;;This next function inserts the statement into the KB if the checks were ok.
(xx%def_fct  ;;xx%def_fct defines a functional relation (if the relation is binary, it is an
             ;; instance of owl:FunctionalProperty and the function is unary)
    xx%assertion_by (?asserter ?assertion_date ?belief_or_def) :-> ?assertion     :=>
  (exist ( (?dated_assertion) (?is_assertable) )              ;;returns nil if no assertion
    (and (= ?dated_assertion
            (if (not (xx%valid_sentence ?belief_or_def))  nil
             (if (or (= ?assertion_date nil) (xx%Definition ?belief_or_def)) ?belief_or_def
              (listof ^(xx%assertion_date ,?belief_or_def ,?assertion_date)) )))
         (= ?is_assertable
            (and (/=  ?dated_assertion  nil)
                 (forall ((?list_of_bad_kinds_of_statement) (?bad_kind_of_statement))
                     (and (xx%kinds_of_statement_this_agent_has_committed_not_to_assert
                                        ?asserter  ?list_of_bad_kinds_of_statement)
                          (item ?list_of_bad_kinds_of_statement ?bad_kind_of_statement)
                          (not (holds ?bad_kind_of_statement ?dated_assertion)) ))))
         (=> ?is_assertable   ;;holds must be used when the predicate/relation is a variable
             (and (wtr ?dated_assertion) ;;wtr: weakly true (asserts if no paradox created)
                  (= ?asserted_belief  ?dated_assertion) ))
         (=  ?assertion  (if (not ?is_assertable)  nil  ?dated_assertion)) )))
```

```
(xx%def_rel xx%Contextualizing_relation (?r) := ;;e.g.: (xx%source '(xx%City xx%Paris) xx)
  (and (relation ?r) ;;actually (xx%Relation_on_a_sentence ?r) but this is redundant here
       (forall ((@args)(?se))
          (=> (holds ?r @args)
              (exists ((?s sentence)) (and (xx%item (listof @args) ?s) (wtr ?s))) ))))

(xx%def_fct xx%atomic_substatements (?sentence) :=      ;;returns contextualized AND-clauses
  (if (not (list ?sentence)) (listof ?sentence)
   (if (= 'and (first ?sentence)) (map xx%atomic_substatements (rest ?sentence))
    (if (xx%Non_contextualizing_relation_on_a_sentence (first ?sentence))
        (append (listof ?sentence) (map first (map xx%atomic_substatements ?sentence)))
     (map first (map xx%atomic_substatements ?sentence)) ))))

;;KIF permits to represent "default rules" (and thereby non-monotonicity and the "closed
;;world assumption") via the use of "=>>" (instead of "=>") and "(consis ?s)" (meaning
;;"as long as ?s is consistent with the current rest of the KB"); see the examples next page
(xx%def_rel xx%default_truth (?sentence) := (wtr ^(=>> (consis ,?sentence) ,?sentence)))

(xx%def_rel xx%End-user (?u) :=> (xx%Agent ?u))   ;;e.g.: (xx%End-user xx)
```

Called by xx%assertion_by, the function xx%valid_sentence returns its parameter if it is a well-formed definition or a belief that does not "implicitly contradict" other statements from the same asserter. A belief *implicitly contradicts* another one when i) they are inconsistent if their contextualizations by their sources are removed, and ii) this inconsistency is not explicitly represented by a relation of type xx%correction from one of the statement to the other one. Some subtypes of xx%correction are xx%corrective_specialization and xx%corrective_generalization. Thus, for example, if an agent has entered his belief that "all birds fly", the KBMS should then not allow him to assert that he believes that "in 2012, between 50% and 75% of Australian birds can fly", *unless* the newer statement also asserts that this new belief corrects the earlier one. With this last example, the KBMS may also detect that the new belief specializes the first and thus may require the use of a relation of type xx%corrective_specialization.

Depending on the KBMS policy, additional constraints may be checked via the last above cited function or the list returned by the function xx%kinds_of_statement_this_agent_has_committed_not_to_assert. For the KB to be what can be called "at least minimally well-organized", the above proposed *"no entering of implicit contradiction" policy* should be extended to also i) take into account beliefs from all sources, not just those from the same source, and ii) prevent the entering of redundancies. The resulting KB organization may then be exploited for automatically choosing between contradictory beliefs. E.g., an agent may specify that, by default, he believes the most specialized corrections from certain kinds of agents. This organization also leads knowledge providers to precise their knowledge objects, thus not forcing other agents to second-guess them.

Some additional definitions and default rules are necessary to help each agent *specify in a flexible way* what he believes in, his quality measures, his notions of "better" for or between various kinds of objects, and how these measures should combine. The flexibility comes from the complex but automatic selections, combinations and overrides allowed by the writing of many simple default rules. For them to work well, the beliefs of the users should not be implicitly contradictory. The next examples show i) a relation to test the non-contradiction of a user's beliefs, ii) a way for the KBMS to start making inferences on the beliefs of the end-user by stating that they are true without their contextualization by their source, iii) a definition and then a rule for the KBMS to state that by default all

agents believe anything as long as they have no reason not to, iv) a definition and then a rule to state that by default, when an agent appears to believe in xx%correction relations between statements, he believes only the most corrected statement, v) a definition to express that an object is "superior to another one for a certain characteristic" if and only if there is a greater-than relationship between the respective recorded values for this characteristic, vi) a rule stating that by default "being superior to another object on a certain characteristic" is being "better than this object for this characteristic" (this rule relies on a particular normalization of the representations), and vii) a definition and then a rule to state that by default, if an object is "better than another one for all its relations", it is "better than this other object". Any user can override these rules by stating his belief in more specialized (or more general) default rules. If definitions were used instead of default rules, the possibilities of overrides and unforeseen combinations would be reduced. These examples of default rules give an idea of what the model permits but are not part of it since a KBMS can choose other default rules.

Via the combination of such rules, object comparisons (via relations such as xx%better_for or xx%superior_for) or evaluations (via numerical values) can be derived from simpler ones. E.g., a function to evaluate the quality of a knowledge management technique can be based on the quality of each input and output that the technique allows. Similarly, the evaluation of tools and knowledge providers can be based on the quality of their knowledge and provided/authored techniques. The current research works on knowledge quality measures (e.g., [17-18] [22-25]; see the next sections) focus on providing simple measures, not on easing their design and combination.

```
(xx%def_rel xx%Believer_of_a_consistent_set_of_beliefs (?agent) :=
  (consis   (forall ((?s sentence)) (=> (xx%believer ?s ?agent) (wtr ?s)))   ) )

(forall ((?a xx%agent))
  (=> (xx%End-user ?a) (xx%default_truth ^(forall ((?s)) (=> (xx%believer ?s ?a) (wtr ?s)))) ) )

(xx%def_rel xx%credulous (?agent) :=  ;;believes anything as long as he has no reason not to
  (forall ((?s)) (xx%default_truth ^(xx%believer (xx%atomic_substatements ,?s) ,?agent))) )
(forall ((?agent xx%agent)) (xx%default_truth ^(xx%credulous ,?agent)) )

(xx%def_rel xx%believer_of_the_most_corrected_version (?agent) :=  (forall ((?s1) (?s2))
  (xx%default_truth (and (xx%believer ^(xx%correction ,?s1 ,?s2) ,?agent)
                         (not (xx%believer ,?s1 ,?agent))  (xx%believer ?s2 ,?agent) ))))
    ;;=> if there is a hierarchy of corrections, only the most corrected is finally believed
(forall ((?a xx%agent) (xx%default_truth ^(xx%believer_of_the_most_corrected_version ,?a)) )

(xx%def_rel xx%superior_for (?chrc ?o1 ?o2) :=   ;;e.g.: (xx%superior_for precision RDF OWL)
  (forall ((?v1) (?v2)) (=> (and (holds ?chrc ?o1 ?v1) (holds ?chrc ?o2 ?v2))
                            (< ?v1 ?v2) )))
(forall ((?a xx%agent) (?chrc) (?o1) (?o2)) (xx%default_truth
  '(xx%believer '(=> (xx%superior_for ?chrc ?o1 ?o2) (xx%better_for ?chrc ?o1 ?o2)) ?a) ) )

(xx%def_rel  xx%better_for_all_relations (?o1 ?o2) :=
  (forall ((?rel)) (xx%better_for ?rel ?o1 ?o2)) )
(forall ((?a xx%agent) (?chrc) (?o1) (?o2))
  (xx%default_truth '(=> (xx%better_for_all_relations ?o1 ?o2) (xx%better ?o1 ?o2))) )
```

3 Description Content Quality

The goal of the top-level ontology presented in the remaining sections is to organize the *main kinds* of methodological elements, best practices, quality characteristics (e.g., evaluation criteria, quality dimensions, the "data quality indicators" of [22], ...) and quality measures (e.g., the "scoring functions" and "assessment metrics" of [22], ...) that have been proposed for knowledge sharing purposes. This article only shows important elements of a *subtype hierarchy of quality measuring functions* on information objects, with the function result being a value (typically, numerical or boolean). Indeed, *relations* can then be derived from boolean functions (the next page includes an example) and, *from this subtype hierarchy*, other ones can be derived, e.g., the one for *quality characteristics* and the one for *"statements that have a certain (kind of) quality measure"* (alias, "statements that follow a certain (kind of) best practice"). This last hierarchy may be proposed to users for them to select "kinds of statements they commit not to assert".

There are many ways to categorize quality evaluations, e.g., according to what kind of object they evaluate, and whether or not they take into account certain lexical, structural or semantic best practices. The next indented list shows one intuitive top-level categorization.

In all such indented lists below, the XML namespace shortcut is used but "xx:" is left implicit. "LDpattern:" is for [18], "LD:" for [20], "SF:" for [21] and "PD:" for [22]. C++/Java-like comments are used. Relation identifiers use nominal expressions and follow the common "graph reading convention", i.e., the last argument is the destination of the relation; thus, a binary relation R(X,Y) can be read "<X> has for <R> <Y>". For functional relations, the last argument is the function result.

```
quality  //function on an object with possibly other arguments; returns a value
  content_based_quality  //at least based on the object content
  meta-statement_based_quality  //at least based on a meta-statement on the object
    rating_based_quality  //at least based on meta-statements that are ratings
```

For each kind of evaluated source object, there are various ways to categorize i) functions that evaluate certain aspects of this kind of objects, ii) functions that evaluate "related objects", and iii) functions that differently aggregate the values returned by these functions. Hence the following subtypes for xx:description_content_quality:

```
description-content_quality  //subtype of the above function xx:quality
  correctness  //one main kind of description-content_quality; the next page gives subtypes
  conformity  //another main kind; the page after the next one gives some important subtypes
  quality_of_this_description_content  //to evaluate the source object on all its criteria
    description_content_quality_of_this_description_content  //content-related aggregations
    quality_of_the_description_media_related_to_this_description_content
    quality_of_the_description_containers_related_to_this_description_content
```

Here, "related" refers to *actual or potential/allowed* relations. E.g., RDF (a description content) *allows* various kinds of textual or graphical notations (description media) – some being standards, some not – even if most RDF-based tools (description containers) only work with RDF/XML. Thus, some evaluation functions may better rate an RDF-based tool that can handle more notations, for example by calling external translation tools.

One handy partition for the description-content_semantic-quality functions is the distinction between those that give "correctness" values for the evaluated object and those checking that it includes certain things. Here are some subtypes for the first kind.

```
correctness  //of the evaluated object (statement or term referring to a statement)
  LD:accuracy  //factual correctness of a statement (which should be a belief) wrt the world
  consistency  //reports all or some inconsistencies and implicit contradictions
    consistency_of_this_statement_wrt_this_one (ST,ST -> boolean) //this signature states
       //that this function is boolean and has exactly 2 statements as arguments;
       //one relation derivable from it is:  xx:statement_consistent_with_this_one (ST,ST)
      consistency_and_non-redundancy_of_this_statement_wrt_this_one  //inherited signature
    consistency_of_this_KB (ST -> boolean)
       consistency_of_the_RDF_KB
         consistency_of_SKOS_relations  //the measures of [24] are subtypes of this type
           consistency_of_an_RDF_KB_tested_via_a_SPARQL_query  //as in [23] and [25]
       LD:internal_consistency_fct  //SF:consistency_fct seems to be an alias
       LD:modeling_correctness_fct //tests if the "logical structure of the data is correct"
          //LD and SF do not precise if these last 2 types are dimensions or functions; this
          // is why "_fct" is added here; other relations and dimensions can be derived
    substatements_of_this_1st_statement_inconsistent_with_this_2nd_one (ST,ST -> set)
    consistency_ratio  //no restriction on the arguments but the result is a number (ratio)
      consistency_ratio_of_such_a_statement_in_this_statement (ST,ST -> number)
      consistency_ratio_of_all_atomic_substatements_in_this_statement (ST -> number)
        consistency_ratio_of_this_KB (ST -> number)
      consistency_ratio_of_relations_on_this_term_in_this_statement (term -> number)
      consistency_ratio_of_this_relation_on_this_term_in_this_statement (ST,term -> number)
        PD:consistency  //"number of non-conflicting frames" divided by "number of frames"
```

All current quality measures related to Linked Data seem to use the whole KB (data set) as *implicit argument* (this eases their use but this is a loss of generality) and most only work on "frames" ("objects" in object-oriented approaches), i.e, on relations from a term. They do not work on *any kind of statement*. The above hierarchy shows how different but related evaluation functions and relations can be organized and generalized. Concept types can be derived too. An important one is xx:Statement_consistent_and_non-redundant_ with_any_other_one_in_the_KB since if a KBMS checks that each statement is of this type before inserting it into the KB, this one will be "at least minimally well-organized". As shown in Section 2, this eases or permits complex evaluations. The above functions are relatively easy to write in KIF. E.g.:

```
(xx%def_fct xx%consistency_ratio_of_such_a_statement_in_this_statement (?s1 ?s2)
    ;;as in all evaluation functions, the first argument is the evaluated object
  (div (xx%cardinality (setofall ?s (and (xx%substatement ?s2 ?s)  (=> ?s ?s1)
                                         (not (=> ?s2 (not ?s))) )))
       (xx%cardinality (xx%substatements ?s2)) ))          ;;?s2 may be a whole KB
```

The next page organizes functions checking that within an object certain elements exist and are conform to a certain pattern. The various subtypes are semantically close. The first presented subtype can be re-used to write the other ones. This specialization hierarchy shows that the current categorizations for Linked Data quality criteria and measures only cover particular cases. Thus, the current implementations of (some of) these measures also only cover particular cases. [23] and [25] proposes such implementations via SPARQL queries and SPIN rules. To save space, there is no repetition of types in this hierarchy (this applies in the next hierarchies too). Some of the types could clearly also appear at other places. The comments give some explanations for each of the types. The ones in bold and/or italics are the most important for categorization or re-use purposes.

conformity //reports on the existence/number of certain things/patterns
conformity_of_this_statement_wrt_this_requirement (ST,ST -> boolean)
ratio_of_conformity_to_this_requirement_in_this_statement (ST,ST -> number)
ratio_of_conformity_of_the_KB //no restriction on the arguments
LD:modeling_granularity (-> number) //no argument
PD:structuredness //e.g., PD:coverage (number of objects with all relations of a schema)
//and PD:coherence (average of coverage for all terms)
PD:completeness //alias, LD:completeness (do all required terms/relations exist?)
PD:intensional_completeness //ratio (percentage) of required relations in the KB
PD:extensional_completeness //ratio of required terms in the KB
PD:LDS_Completeness //ratio of terms with a certain relation (property) in the KB
PD:relevancy //alias LD:Boundedness, ratio of relevant data for an application
PD:verifiability //existence of information to check for correctness; subtype examples:
//PD:traceability, PD:provability, PD:accountability; the following best practices are
//related to these subtypes: "providing another KB for tools that cannot perform
//complex inferences" (LDpattern:materializing_inferences) and "transforming the KB to
//conform to some models" (LDpattern:transformation_query)
SF:validity //no syntax errors, …; very related to PD:verifiability; PD:validity is a subtype
SF:amount_of_data //this function too is a genuine "function subtype" of xx:conformity
representation_quality
organization //e.g., an "at least minimal" one, another one for informal objects too, …
reachability //of the evaluated object (PD:reachability when it is a whole KB)
out-relations //from the object; for a whole KB: PD:external_links, PD:outdegree, …
//the more, the better: this is Berners-Lee's 4th basic rule for Linked Data [2];
//the more widely known/deployed the target objects, the better
in-relations //to the object; for a KB: PD:indegree, PD:LDSInDegreeReachability, …
non-redundancy //e.g., PD:conciseness, PD:intensional_conciseness, …
expressiveness_economy //avoidance of expressive constructs when this does not
//bias knowledge representation and reduce knowledge matching/inferencing/readability
modeling_uniformity //e.g., checks some lexical/structural/ontological conventions
LD:directionality //checks the consistency in the direction of relations
use_of_the_graph-oriented_reading_convention //also very important for readability
conformity_to_an_abstract_model_or_ontology_or_methodology
conform_to_Ontoclean //checks that the object (or each of its sub-objects) is
//instance of the Ontoclean 2nd-order types: (semi/anti/totally_)rigid_thing, …
use_of_a_standard_model //3rd basic rule for Linked Data for abstract models only
quality_of_the_representation_of_terms
identification_by_properly_formed_URIs //checks that objects are identified by
//HTTP URIs that can be dereferenced (by agents to find further information; the
//first 2 rules of Linked Data [2]; [18] specializes these best practices)
following_of_naming_conventions //use of nouns, of a loss-less naming style, …
LD:referential_correspondence //consistency and non-redundancy of identifiers
LD:typing //checks that nodes are first-order typed entities, not just strings,
// hence checks the "Link Not Label" best practice [18]
PD:vocabulary_understandability //checks that terms have human readable labels, …
LD:intelligibility //alias SF:comprehensibility? They seem to be only about terms
PD:internationalization_understandability //checks that the language is specified
quality_of_existing_or_derivable_relations
use_of_binary_relations_only //since this helps knowledge matching and precision
quality_of_existing_or_derivable_meta-statements //hence relations from statements
quality_of_existing_or_derivable_contexts //temporal/spatial/modal/…
provenance //checks the sources (agents/files) are represented (LD:Attribution)
//and the creation dates too (LD:History); LD:Authoritative is for
//checking if the author is a credible authority on the subject
loss-less_integration //checks that the semantics of source objects was not changed
PD:timeliness //alias SF:timeliness and LD:Currency; is the object is up-to-date?
//E.g., PD:newness (timely creation) and PD:freshness (timely update)
SF:licensing //alias, LD:licensed; to check for an open license, use PD:openness
security //checks for signatures, encryption, maintainability (LD:sustainable), …

4 Description Medium Quality

Description-medium quality functions evaluate the textual/graphic/... presentation of some description content objects in some description containers. The more structured, "distinguished from content" and adaptable by the end-users these media are, the better. There are some tools for adapting the "classic presentation aspects" (fonts, forms, …) of knowledge objects – via the use of CSS or XSLT [28] – but not the (input and output) notations themselves. Here is a specialization hierarchy for description-medium quality evaluating functions. The general comments on the previous hierarchies also apply here.

```
description-medium_quality  //subtype of the function xx:quality
  quality_of_this_description_medium  //to evaluate the source object on all its criteria
    description_medium_quality_of_this_description_medium  //medium-related aggregations
    quality_of_the_description_content_related_to_this_description_medium
    quality_of_the_description_containers_related_to_description_medium
  use_of_standard_formats  //for used KRLs (RDF/XML, …), character encodings, graphics (SVG, …),
    //… (see w3.org); this is the 3rd Linked Data [2] basic rule but for concrete models only
  use_of_structured_formats  //e.g., an HTML presentation (possibly with RDFa statements)
    use_of_formats_distinguishing_structure_from_presentation  //like XML except that it
      //does not permit its users to adapt its notation via the setting of some values
      use_of_notations_that_can_be_adapted_by_the_user  //unlike XML and almost all notations
      use_of_machine-understandable-formats
        use_of_formats_that_have_an_interpretation_in_some_logic
          PD:format_interpretability  //aggregation on qualities of formats proposed by a KB
            PD:human_and_machine_interpretability  //N3 is more easily read than RDF/XML
        format_structural_quality  //e.g., SF:Versatility
          format_abstract-expressiveness  //the expressiveness of its abstract model
            //(→ predicate_logic, first-order_logic, …), kinds of possible quantification
            //(note: KIF allows to define all kinds of relations to represent numerical
            // quantifiers but has no predefined keywords for them; thus, numerical quantifiers
            //defined by different users will be hard to match (especially via simple
            //graph-matching based techniques; hence, KIF is expressive but low-level
          syntactic_expressiveness  //the higher the result, the higher-level the notation
              //can be considered (for the selected criteria), i.e., the more flexible and
              //readable the format is, the more normalized/uniform the descriptions are,
              //and hence the easier to compare via graph-matching these descriptions are
            syntactic_constructs_for_logical_ones  //e.g., are there keywords for numerical
              //quantifiers (and for which kinds, e.g., "58%", "2 to 6") in the format
            syntactic_constructs_for_creating_shortcuts  //kinds of lambda-abstractions, …
            syntactic_constructs_for_ontological_primitives  //e.g., for type partitions and
              //primitives such as those in Ontoclean and extensions of them. They are needed
              //for knowledge engineering [3]. RDF is low-level: it has no keywords for them
              //but can import a language ontology which has them
            referable_first-order-entities  //e.g., what can be a 1st-order entity, i.e.,
              //what can be referred to via a variable in the notation: concept nodes,
              //relation nodes, quantifiers, …; the more things can be 1st-order entities
              //(and hence that can be related to other things, annotated, selected via a
              //mouse, …), the better, and the more freely and formally related, the better
              //for structuring/annotation flexibility purposes; from that viewpoint, an
              //interface/notation for a KB may be better than one for a database or a
              //structured document (which is then also better than an unstructured one)
  format_concision  //e.g., N3 is more concise than RDF/XML
  format_uniformity //reports on the extent to which similar things can be (re)presented in
                    //similar ways (from a software viewpoint and/or from a person viewpoint)
    SF:Uniformity  //xx:format_uniformity for a whole KB
  performance_of_this_format_for_this_task (description_medium, task -> value)
```

5 Description Container Quality

Description-containers quality functions evaluate the way a given description container (static file, distributed or not KB server) modularizes, stores, makes retrievable and accessible (i.e., how it "publishes") description content objects and checks or allows updates or queries on them. Compared to the independent and direct use of static files (e.g., RDF files), the use of knowledge servers by people eases knowledge modeling and reduces the implicit inconsistencies and redundancies between their knowledge statements. A server can also use static input/output files and offers much more flexibility than static files. It can also provide more services than those of a description-container (e.g., it can forward queries). This can be taken into account for evaluating its quality. Here is a specialization hierarchy for description-container quality evaluating functions. The general comments on the previous hierarchies still apply.

```
description-container_quality  //subtype of the function xx:quality
  quality_of_this_description_container  //to evaluate the source object on all its criteria
    description_container_quality_of_this_description_container  //sub-quality aggregation
    quality_of_the_description_content_related_to_this_description_container
    quality_of_the_description_media_related_to_description_container
    quality_of_the_processes_supported_by_this_description_container
  storage_related_quality
    maximal_size_of_the_KB
    container_based_modularization
      static_container_based_modularization  //static file based modules/versions/...
      dynamic_container_based_modularization  //forwarding of knowledge/queries amongst KBs
      LD:connectedness  //checks if combined datasets join at the right points
  assertion_related_quality  //what can be added or updated, by whom, in which language, …
    ontological_flexibility  //is the ontology fixed, i.e., is the KB actually a database?
      LDpattern:annotation  //are third-party resources accepted?
      LDpattern:progressive_enrichment  //ways data (model) can be improved over time
    checking_possibilities  //what kinds of inconsistencies or redundancies can be detected?
       //does the server advocate best practices to its users?
  information_retrieval_related_quality  //on the whole KB or n some of its statements
    published_or_given_metadata  //on the KB or a art of it, e.g.,
      //via just a "topic" (Ldpattern:Document_Type), via the use of a semantic sitemap [11]
      //via voiD (Vocabulary of Interlinked Datasets) [12], via DCAT [13], via metadata given
      //for any object (if a user requests it) but calculated in a predefined way (as with
      //"Concise Bounded Descriptions") [2], or via metadata accessible via powerful queries
    object_accessibility
      PD:accessibility  //access methods, e.g., via SPARQL, an API, a file (HTML,RDF)
      PD:availability  //percentage of time a given service is "up"
      SF:performance  //low latency, high throughput, only minor "performance variations", …
        PD:response_time  //e.g., for static access and SPARQL access
        PD:robustness  //average of performance over time; caching data and using
                       // LDpattern:parallel_loading help performance
      querying_possibilities  //what can be queried, with which input/ouput languages,
         //what privacy techniques are used, are the results ranked, filtered and merged, …
  interface_personalization  //to which extent can the input/output presentation be adapted
      //by end-users and can take into account their constraints: language, disabilities,
      //access from various devices (mobile ones, …), from various software (browsers, …), …
```

6 Some Other Categorizations

In order to show how this knowledge criteria/quality ontology extends, generalizes and organizes the elements of its sources, the next indented lists show the structure of the four most organized sources that so far seemed to exist for Linked Data, even though they are essentially only two levels deep. "SF:" is for [21], "Kahn" is for [19], "LDpattern:" is for [18] and "OPD:" is for [17] (this last source has three 3-level deep categories and one 4-level deep category). The first two sources are for quality criteria, the last two are for best practices. Their categories – the ones shown below – *seem to be* concept types. To permit a maximal integration of the various sources, they have been integrated into this quality ontology via function types, as illustrated by the previous indented lists. From these functions hierarchies, the concept types hierarchies can be generated. In the following lists, the lowermost categories are given within comments *and* without prefix for their source. The lowermost "OPD" subtypes have several instances in the OPD library.

```
SF:Quality_criterion //this is the categorization that is closest to the decomposition
                     //according to description content/medium/container
  SF:Content  //Consistency, Timeliness, Verifiability
  SF:Representation  //Uniformity, Versatility, Comprehensibility
                     //mixes criteria on description medium and description container
  SF:Usage  //Validity of documents, Amount ofData, Licencing
            //mixes criteria on description content and description container
  SF:System  //Accessibility, Performance

Kahn:Quality_dimension //these are "the 15 most important ones from consumer perspective"
  Kahn:Intrinsic  //Believability, Accuracy, Objectivity, Reputation
  Kahn:Contextual  //Value-added, Relevancy, Timeliness, Completeness, Appropriate amount
  Kahn:Representational  //Interpretability, Ease of understanding, consistency, Concision
  Kahn:Accessibility  //Accessibility,  Access security

LDpattern:Linked_Data_pattern
  LDpattern:Identifier_pattern  //Hierarchical URIs, Literal Keys, Natural Keys,
                                //Patterned URIs, Proxy URIs, Shared Keys , URL Slug
  LDpattern:Modelling_pattern  //Custom Datatype, Index Resources, Label Everything,
    //Link Not Label, Multi-Lingual Literal, N-Ary Relation , Ordered List,
    //Ordering Relation, Preferred Label , Qualified Relation, Reified Statement,
    //Repeated Property, Topic Relation, Typed Literal
  LDpattern:Publishing_pattern  //Annotation, Autodiscovery, Document Type, Edit Trail,
    //Embedded Metadata, Equivalence Links, Link Base, Materialize Inferences,
    //Named Graphs, Primary Topic,  Autodiscovery, Progressive Enrichment, See Also
  LDpattern:Application_pattern  //Assertion Query, Blackboard, Bounded Description,
    //Composite Descriptions, Follow Your Nose, Missing Isn't Broken, Parallel Loading,
    //Parallel Retrieval, Resource Caching, Schema Annotation, Smushing, Transformation Query

ODP:Ontology_Design_Pattern
  ODP:Structural ODP  //Architectural ODP, ODP:Logical ODP
      ODP:Logical ODP  //Logical_macro_ODP, Transformation_ODP
  ODP:Correspondence ODP  //Alignment ODP, Re-engineering ODP
    ODP:Re-engineering ODP  //ODP:Schema_reengineering_ODP
      ODP:Schema_reengineering_ODP
  ODP:Content ODP,  ODP:Reasoning ODP,  ODP:Lexico-Syntactic ODP
  ODP:Presentation ODP  //Naming ODP, Annotation ODP
```

7 Conclusion

This article has presented the top-level of an ontology organizing knowledge sharing best practices, design patterns, evaluation criteria and evaluation measures in a systematic, non-redundant and scalable way (e.g., by being based on distinctions on information objects rather than on processes). Some other research works on this subject mainly proposed *lists* of categories with, sometimes, some implementations (e.g., via SPARQL). This work shows that these categories and implementations are only particular cases which could sometimes be easily generalized. It also permits to have a more synthetic view of the *kinds* of things that could or should be evaluated or done during knowledge sharing, or proposed by knowledge engineering/sharing tools. This ontology can be extended by Web users via the server which hosts it [xx]. This ontology could then be used as an index for elements of other libraries or ontologies. To that end, the bigger it will become, the more useful it will be.

Section 2 introduced the formalization of a model enabling the integration of knowledge from various sources into a consistent and "at least minimally well organized KB" which eases the filtering and evaluation of knowledge. Section 2 also illustrated the use of "default rules" to allow the combination of simple evaluation functions into complex ones and the re-use of other agents' functions. The distinction made between the statements that can be contradicted and those which cannot seems to be a useful element for knowledge editing/integration protocols to handle cooperation between agents and encourage them to give the meta-statements necessary to evaluate knowledge and the providers of this knowledge. The fact that knowledge on the Semantic Web is full of implicit contradictions and redundancies, very hard to evaluate, and often incorrect with respect to the OWL primitives that it re-uses [27], may be an indication that such protocols and kinds of evaluations are useful for the Semantic Web.

References

1. Palma, A., Haase, P., Wang, Y., d'Aquin, M.: D1.3.1 propagation models and strategies. Technical report, NeOn Deliverable D1.3.1 (2007)
2. Heath, T., Bizer C.: Linked Data: Evolving the Web into a Global Data Space. Synthesis Lectures on the Semantic Web: Theory and Technology, 1:1, pp. 1–136. Morgan & Claypool (2011)
3. Guizzardi, G., Lopes, M., Baião, F., Falbo, R.: On the importance of truly ontological representation languages, IJISMD, 2010. ISSN: 1947-8186
4. Patel-Schneider, P.F.: A Revised Architecture for Semantic Web Reasoning. In: PPSWR 2005, LNCS, vol. 3703, pp. 32–36 (2005)
5. Genesereth, M.R., Fikes, R.E.: Knowledge Interchange Format, Version 3.0. Reference Manual. Technical Report Logic-92-1, Stanford University (1992)
6. Genesereth, M.R.: Knowledge Interchange Format. Draft proposed American National Standard (dpANS), NCITS.T2/98-004 (1998)
7. Hayes, P., Menzel, C., Sowa, J., Tammet, T., Altheim, M., Delugach, H., Gruninger, M.: Common Logic (CL): a framework for a family of logic-based languages. ISO/IEC IS 24707:2007

8. Farquhar, A., Fikes, R., Rice, J.: The Ontolingua Server: a Tool for Collaborative Ontology Construction. International Journal of Human-Computer Studies (1996)
9. Borgo, S., Masolo, C.: Ontological Foundations of DOLCE. Handbook on Ontologies, Springer, pp. 361–382 (2009)
10. Guarino, N., Welty, C.: Evaluating Ontological Decisions with OntoClean. Communications of the ACM, vol. 45(2), pp. 61–65 (2002)
11. Cyganiak, R., Delbru, R., Stenzhorn, H., Tummarello, G., Decker, S.: Semantic sitemaps: Efficient and flexible access to datasets on the semantic web. In: 5th European Semantic Web Conference (2008)
12. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J. Describing linked datasets. In: WWW2009 Workshop on Linked Data on the Web (2009)
13. Maali, F., Erikson, J., Archer, P., DCAT Catalog Vocabulary. W3C Working Draft (2012)
14. Breuker, J., van de Velde, W. : CommonKADS Library for Expertise Modelling: Reusable Problem Solving Components. IOS Press (1994)
15. Dromey, G.: Scaleable Formalization of Imperfect Knowledge. In: AWCVS, 2006, pp. 21–33. (2006).
16. Pan, J.Z., Lancieri, L., Maynard, D., Gandon, F., Cuel, R., Leger, A.: Success Stories and Best Practices. Deliverable D1.4.2v2 of KWEB (Knowledge Web), EU-IST-2004-507482
17. Presutti V., Gangemi, A.: Content Ontology Design Patterns as practical building blocks for web ontologies. In: ER 2008, Spain (2008) see http://ontologydesignpatterns.org
18. Dodds L., Davis I.: Linked Data Patterns – A pattern catalogue for modelling, publishing, and consuming Linked Data, http://patterns.dataincubator.org/book/, 56 pages (2011)
19. Kahn, B. K., Strong, D. M., Wang, R. Y.: Information quality benchmarks: product and service performance. Communications of the ACM, vol. 45(4), pp. 184–192 (2002)
20. Mcdonald, G.: Quality Indicators for Linked Data Datasets. http://answers.semanticweb.com/questions/1072/quality-indicators-for-linked-data-datasets (2011)
21. Flemming A., Hartig, O.: Quality Criteria for Linked Data Sources http://sourceforge.net/apps/mediawiki/trdf/index.php?title=Quality_Criteria_for_Linked_Data_sources (2010)
22. Mendes, P.N., Bizer, C., Young J.H., Miklos, Z., Calbimonte J.P., Moraru, A.: Conceptual model and best practices for high-quality metadata. Delivery 2.1 of PlanetData, FP7 project 257641 (2012)
23. Bizer, C.: Quality-Driven Information Filtering in the Context of Web-Based Information Systems. PhD dissertation (195 pages), Free University of Berlin, (2007)
24. Mader, C.: Quality Criteria for SKOS Vocabularies https://github.com/cmader/qSKOS/wiki/Quality-Criteria-for-SKOS-Vocabularies (2012)
25. Fürber, C.: Data Quality Constraints Library. http://semwebquality.org/documentation/primer/20101124/ (2010)
26. Gómez-Pérez, A., Ciravegna, F.: SEALS EU infrastructures project – semantic tool benchmarking. http://www.seals-project.eu/ (2012)
27. Hogan, A., Harthy, A., Passanty, A. Deckery, S., Polleres, A.: Weaving the Pedantic Web. In: LDOW 2010
28. Pietriga E., Bizer C., Karger D., Lee R. (2006). Fresnel: A Browser-Independent Presentation Vocabulary for RDF. In: ISWC 2006, LNCS, vol. 4273, pp. 158–171 (2006)
29. xx
30. xx